

PATENT SPECIFICATION

(11) 1 444 141

1 444 141

- (21) Application No. 32269/73 (22) Filed 6 July 1973
(31) Convention Application No. 269 899
(32) Filed 7 July 1972 in
(33) United States of America (US)
(44) Complete Specification published 28 July 1976
(51) INT CL² G06F 3/02
(52) Index at acceptance

G4A 10EX 12N 12P 15A2 16G 17B4 17P 2HX 3N 3W10W2
3W2 3W4X 5A 5B 6G 6H 6MX 8C 9C 9X AP KB
PX SX



(54) IMPROVED PROGRAMMABLE CALCULATOR

(71) We, HEWLETT-PACKARD COMPANY OF 1501 Page Mill Road, Palo Alto, California, United States of America, a corporation organised and existing under the laws of the State of California, United States of America, do hereby declare the invention, for which we pray that a Patent may be granted to us and the method by which it is to be performed, to be particularly described in and by the following statement:—

This invention is concerned with improvements in or relating to calculators and is more especially concerned with programmable calculators that may be controlled both manually from a keyboard input unit and automatically by a stored program loaded into the calculator from the keyboard input unit or an external record member.

Computational problems may be solved manually, with the aid of a calculator (a dedicated computational keyboard-driven machine that may be either programmable or nonprogrammable), or a general purpose computer. Manual solution of computational problems is often very slow, so slow in many cases as to be an impractical, expensive, and ineffective use of the human resource, particularly when there are other alternatives for solution of the computational problems.

Non-programmable calculators may be employed to solve many relatively simple computational problems more efficiently than they could be solved by manual methods. However, the keyboard operations or language employed by these calculators is typically trivial in structure, thereby requiring many keyboard operations to solve more general arithmetic problems. Programmable calculators may be employed to solve many additional computational problems at rates hundreds of times faster than manual methods. However, the keyboard language employed by these calculators is also typically relatively simple in structure, thereby again requiring many keyboard operations to solve more general arithmetic problems.

Another basic problem with nearly all of the keyboard languages employed by conventional programmable and nonprogrammable calculators is that they allow the characteristics of the hardware of the calculator to show through to the user. Thus, the user must generally work with data movement at the hardware level, for example, by making sure that data is in certain storage registers before specifying the operations to be performed with that data and by performing other such "housekeeping" functions. In addition, these languages have been unique to a particular calculator and not ones generally familiar to those persons skilled in the computer and calculator arts.

In the past both programmable and nonprogrammable calculators have generally had very limited memories, thereby severely limiting the size of the computational problems they could be employed to solve. Because of these limitations, the relatively simple structure of the keyboard languages employed by these calculators and the "housekeeping" requirements associated with their languages have not heretofore been serious shortcomings. However, with advances in technology, the cost of memories has decreased to a point where larger memories could be economically included in programmable calculators. These larger memories have allowed larger and larger problems to be handled by programmable calculators. As a result the shortcomings of conventional calculator languages have become more critical, thereby creating the need for higher level keyboard languages.

(14) In addition to the foregoing shortcomings, conventional programmable calculators generally have less capability and flexibility than is required to meet the needs of many users. For example, they typically cannot be readily expanded and adapted by the user to increase the amount of program and data storage memory or to perform many special keyboard functions oriented toward the environment of the user.

In some conventional programmable calculators a program stored within the calculator can be recorded onto an external magnetic record member and can later be reloaded back into the calculator from the magnetic record member. However, data and programs stored within these calculators typically cannot be separately recorded onto an external magnetic record member and later separately reloaded back into the calculator therefrom. Moreover, these calculators typically have no provision for making a program secure when it is recorded onto an external magnetic record member. Any user may therefore re-record the program or obtain an indication of the individual program steps once the program is reloaded into the calculator.

Conventional programmable calculators with self-contained output display units typically have little or no alpha capability and typically can only display the contents of one or more selected registers. They are therefore typically unable to display a line containing an alpha-numeric statement or an alphabetic message such as might be used, for example, to inform the user how to run programs with which he may be unfamiliar. Such features would be very helpful to the user both in editing programs and in simplifying their use.

Conventional programmable calculators typically have little or no capability for editing keyboard entries or programs stored within the calculator. For example, they typically have no provision for deleting, replacing, and inserting information included in or omitted from a keyboard entry or internally-stored program on a character-by-character or line-by-line basis. As another example, they typically have no provision for directly recalling any line of an internally-stored program. As a further example, they typically have no provision for automatically accommodating and sequencing program statements which are entered by the user in random order. Such features would be very helpful to the user in editing programs.

Conventional computers typically pose an interfacing problem between the user and the machine. This interface requirement takes the form of a machine-level operator with special abilities for maintaining the software system in operative condition for the user. Computer time sharing systems comprising a centrally located computer and a multiplicity of remotely located user-terminals connected thereto by telephone lines have partially solved the user/machine interface problem. However, these systems lack the same flexibility as conventional computers in that they are only programmable and provide no convenient non-programmable method for performing relatively simple calculations. Both types of systems lack provision for editing a program statement from a keyboard without the necessity of retyping the entire statement.

The principal object of this invention is to provide an improved programmable calculator that has more capability and flexibility than conventional programmable calculators, that is smaller, less expensive and more efficient in calculating elementary mathematical functions than conventional computer systems, and that is easier to utilize than conventional programmable calculators or computer systems.

Another object of this invention is to provide a calculator employing BASIC computer language implemented in a read-only memory (ROM) that completely eliminates the user/machine interface requirement of conventional computers and further eliminates the necessity of learning a non-universal language such as those typically associated with conventional programmable calculators.

The present invention provides an electronic calculator comprising keyboard input means including a plurality of alphameric keys for entering one or more BASIC language statements into the calculator, an execute control key for initiating execution of a BASIC language statement entered into the calculator, and a store control key for initiating storage of a BASIC language statement entered into the calculator; memory means, coupled to said keyboard input means, for storing a program of BASIC language statements entered into the calculator; processing means coupled to said keyboard input means and said memory means, said processing means being responsive to actuation of said execute control key, following entry of a BASIC language statement into the

15

calculator, for executing that BASIC language statement; said processing means being further responsive to actuation of said store control key, following entry of a BASIC language statement into the calculator, for storing that BASIC language statement as part of a program in said memory means; and output means, coupled to said memory means, for providing an output indication of the result of execution of a BASIC language statement.

In a calculator as set forth in the last preceding paragraph preferably said processing means includes a read-only memory storing a plurality of routines and subroutines to be selectively performed by the calculator in executing and storing one or more BASIC language statements; said processing means is responsive to actuation of said execute control key, following entry of a BASIC language statement into the calculator, for selectively performing one or more of the routines and subroutines stored in said read-only memory to execute that BASIC language statement; and said processing means is responsive to actuation of said store control key, following entry of a BASIC language statement into the calculator, for selectively performing one or more of the routines and subroutines stored in said read-only memory to store that BASIC language statement in said memory means.

In a calculator as set forth in the last preceding paragraph or the last preceding paragraph but one preferably said keyboard input means includes a space key; and said processing means is responsive to actuation of said space key during entry of a BASIC language statement into the calculator for introducing a blank character at selected locations within that BASIC language statement.

In a calculator as set forth in any one of the last three immediately preceding paragraphs said output means may comprise printing means, coupled to said keyboard input means and memory means, for printing lines of one or more alphameric characters each; said keyboard input means including memory listing means for initiating the listing of any designated number of lines of one or more alphameric characters each then stored in said memory means; and said processing means also being coupled to said output printing means for causing said output printing means to selectively print out the designated lines of one or more alphameric characters each in response to actuation of said memory listing means.

In a calculator as set forth in the last preceding paragraph but three said keyboard input means may also comprise a plurality of keys that may be associated with user-definable functions; said memory means can store a program of one or more lines of one or more alphameric characters each as aforesaid and user-definable functions associated with said plurality of keys; said keyboard input means including memory erase means for selectively initiating the erasure of said program and/or said user-definable functions stored in said memory means; and said processing means is capable of selectively erasing a portion or all of said program, any one or all of said user-definable functions, or all of said program and all of said user definable functions as specified by actuation of said memory erase means.

In a calculator as set forth in any one of the last five immediately preceding paragraphs every stored line may be associated with a separate line number; said keyboard input means may include a halt execution key for designating the line number associated with a line at which execution is to be halted; and said calculator further comprises a program counter, coupled to said processing means, for storing the line number associated with the line currently being executed; temporary storage means, coupled to said keyboard input means and memory means, for storing the line number associated with the line at which execution is to be halted; and said processing means includes logic means, coupled to said keyboard input means, program counter, and temporary storage means, and capable of storing the line number associated with the line at which execution is to be halted in said temporary storage means in response to actuation of said halt execution key, followed by actuation of one or more alphameric keys designating that line number, prior to execution of the lines stored in said memory means and also capable of subsequently halting the execution of those lines in response to the occurrence of a condition of equality between the contents of said program counter and said temporary storage means.

In a calculator as set forth in the last preceding paragraph preferably said keyboard input means includes a first program control key; and said logic means is responsive to actuation of said first program control key, followed by actuation of one or more numeric keys designating the line number of a starting line of the

program stored in said memory means, for initiating execution of the program stored in said memory means, at that starting line.

In a calculator as set forth in the last preceding paragraph preferably said keyboard input means includes a second program control key; said memory means can store variables employed in the program stored in said memory means; said logic means is responsive to actuation of said first program control key for setting all of the variables of the program stored in said memory means to an initial state prior to initiating execution of the program; and said logic means is further responsive to actuation of said second program control key, followed by actuation of one or more numeric keys designating the line number of a line in the program stored in said memory means, for initiating execution of the program at that line number without altering the variables stored in said memory means.

In a calculator as set forth in the last preceding paragraph but seven said keyboard input means may include an initialize key; said memory means, coupled to said keyboard input means, being capable of storing a program of one or more lines of one or more alphameric characters each entered into said calculator and of storing variables associated with that program; and said processing means including logic means, coupled to said keyboard input means and memory means, for setting all of the variables of the program stored in said memory means to an initial state in response to actuation of said initialize key.

In a calculator as set forth in the last preceding paragraph preferably said logic means is responsive to entry of an array definition command by actuation of said keyboard input means to store said array definition command in said memory means; and said logic means is responsive to said array definition command, when encountered during a program execution mode of said calculator, for dedicating a portion of said memory means as storage for the previously defined array.

In a calculator as set forth in the last preceding paragraph preferably said logic means is responsive to actuation of said initialize key for executing an array definition command previously stored in said memory means.

In a calculator as set forth in the last preceding paragraph but ten every stored statement may be associated with a different number; said calculator further comprising magnetic reading and recording means, coupled to said memory means, for transferring lines of one or more alphameric characters between said memory means and a magnetic record member, said keyboard input means being operable for entering control commands into the calculator to control the operation of said magnetic reading and recording means; and said processing means includes logic means, coupled to said keyboard input means, memory means, and magnetic reading and recording means, said logic means being responsive to one of said control commands for securing any portion of said program designated by a beginning line number and an ending line number, said logic means thereafter being responsive to another of said control commands for inhibiting transfer of the secured portion of said program from said memory means to a magnetic record member.

In a calculator as set forth in the last preceding paragraph but eleven said keyboard input means may include a second plurality of keys each of which may be associated with a function of one argument, and said processing means including logic means, coupled to said keyboard input means and memory means, for associating a selected function with a selected one of said second plurality of keys in response to sequential actuation of selected ones of the first and said second pluralities of keys designating the selected function and the selected one of said second plurality of keys and for storing the selected function as part of the program stored in said memory means in response to actuation of said store control key.

In a calculator as set forth in the last preceding paragraph preferably said logic means is responsive to actuation of said execute control key, following actuation of a selected one of said second plurality of keys with which a selected function has previously been associated, for initiating execution of that selected function by said processing means.

In an electronic calculator as set forth in the last preceding paragraph but thirteen said keyboard input means may comprise a plurality of user-definable keys each of which may be associated with one or more lines of BASIC language program statements; said memory means being capable of storing lines of BASIC language program statements associated with selected ones of said plurality of definable keys; and said processing means including logic means, coupled to said keyboard input means and memory means, for initiating execution of the BASIC

language program statements associated with a selected one of said definable keys in response to actuation thereof.

In an electronic calculator as set forth in the last preceding paragraph but fourteen said processing means may be capable of defining a square matrix in response to a matrix command entered into the calculator from said keyboard input means, for thereafter storing the matrix elements in said memory means as designated from said keyboard input means, and for thereafter evaluating the determinant of the previously defined square matrix in response to a determinant command entered into the calculator from said keyboard input means.

In an electronic calculator as set forth in the preceding paragraph but fifteen preferably said processing means includes a read-only memory in which routines and subroutines for defining the square matrix, storing the matrix elements, and evaluating the determinant of the defined square matrix, are stored.

An electronic calculator as set forth in the last preceding paragraph but sixteen may further comprise a print control key for designating a print-all mode of calculator operation, said processing means including logic means, coupled to said keyboard input means, memory means, and display means, for initiating the display of lines of one or more alphameric characters entered into the calculator, alphameric error messages generated during execution of lines of one or more alphameric characters by said processing means, and executed display commands; and printing means for printing lines of one or more alphameric characters, said logic means being responsive to actuation of said print control key for initiating the printing of lines of one or more alphameric characters entered into the calculator, error messages generated during execution of lines of one or more alphameric characters by said processing means, and executed display commands.

In an electronic calculator as set forth in the last preceding paragraph but seventeen each of said alphameric keys may also be representative, either alone or in combination with others of said alphameric keys, of an ASCII character; and said calculator may further comprise buffer storage means, coupled to said keyboard input means and to said memory means, for storing a line of alphameric information entered into the calculator; said memory means being also coupled to said buffer storage means for storing a program of one or more lines of alphameric information entered into the calculator; and said processing means being also coupled to said buffer storage means for executing the line of alphameric information then stored in said buffer storage means or a program of one or more lines of alphameric information then stored in said memory means; and said processing means including logic means coupled to said keyboard input means and responsive to actuation of selected ones of said plurality of alphameric keys, singly or in combination, for generating binary codes representative of every ASCII character and for storing those binary codes in said buffer storage means.

In an electronic calculator as set forth in the last preceding paragraph preferably said keyboard input means includes a terminal transmit key; said calculator includes a modem for transmitting and receiving alphameric information over telephone lines; and said logic means is responsive to actuation of said terminal transmit key for transmitting through said modem the ASCII codes represented by the binary codes then stored in said buffer storage means.

An electronic calculator as set forth in the last preceding paragraph but nineteen may further comprise: input/output interface means, including a plurality of input/output channels, for coupling selected peripheral input/output units to the calculator; and said processing means may include logic means coupled to said keyboard input means, memory means, and input/output interface means, said logic means being responsive to execution of a binary output statement, including a base ten number and entered from said keyboard input means or stored in said memory means, for converting said base ten number to a corresponding twelve-bit binary code and for transmitting said twelve-bit binary code to a selected one of said input/output channels.

An electronic calculator as set forth in the last preceding paragraph but twenty may further comprise: input/output interface means, including a plurality of input/output channels, for coupling selected peripheral input/output units to the calculator; and said processing means may include logic means coupled to said keyboard input means, memory means, and input/output interface means, said logic means being responsive to execution of a binary read statement, entered from said keyboard input means or stored in said memory means, for reading an eight-bit word on a selected one of said input/output channels, for converting said

18

6

1,444,141

6

eight-bit word to a base ten number, and for associating said base ten number with a designated variable.

5 An electronic calculator as set forth in the last preceding paragraph but twenty-one may further comprise: input/output interface means, including a plurality of input/output channels, for coupling selected peripheral input/output units to the calculator; and said processing means may include logic means coupled to said keyboard input means, memory means, and input/output interface means, said logic means being responsive to execution of a status statement, entered from said keyboard input means or stored in said memory means, for interrogating a selected input/output channel to obtain an indication of the operating state of a selected peripheral input/output unit associated therewith.

10 In an electronic calculator as set forth in the last preceding paragraph preferably said logic means is operable for generating a number indicative of the operating state of the interrogated peripheral input/output unit.

15 In an electronic calculator as set forth in the last preceding paragraph but twenty-three said processing means may include logic means coupled to said keyboard input means and memory means, said logic means being responsive to execution of a rotate statement, entered from said keyboard input means or stored in said memory means, for rotating a specified number of the individual bits of a designated sixteen-bit integer word entered into the calculator or stored in said memory means.

20 In an electronic calculator as set forth in the last preceding paragraph preferably said logic means is responsive to execution of an AND statement for performing, according to the rules of Boolean algebra, the logical AND operation on the corresponding bits of two designated sixteen-bit integer words entered into the calculator or stored in said memory means and for associating the resulting sixteen-bit word with a specified variable.

25 In an electronic calculator as set forth in the last preceding paragraph but one preferably said logic means is responsive to execution of an OR statement for performing, according to the rules of Boolean algebra, the logical OR operation on the corresponding bits of two designated sixteen-bit integer words entered into the calculator or stored in said memory means and for associating the resulting sixteen-bit word with a specified variable.

30 In an electronic calculator as set forth in the last preceding paragraph but twenty-six said keyboard input means may employ ASCII coding; and said calculator may further comprise input/output interface means, including a plurality of input/output channels, for coupling selected peripheral input/output units to the calculator; and said processing means includes logic means coupled to said keyboard input means, memory means, and input/output interface means, said logic means being responsive to execution of an enter statement, entered from said keyboard input means or stored in said memory means, for receiving information from a selected peripheral input/output unit and for converting the received information to ASCII code when it is received from a peripheral input/output unit employing a non-ASCII code, said logic means being further responsive to execution of an output statement, entered from said keyboard input means or stored in said memory means, for transmitting information from the calculator to a selected peripheral input/output unit and for converting such information from ASCII code to any non-ASCII code employed by the selected peripheral input/output unit.

35 In the complete specification of our copending U.K. patent application No. 3345/76 Serial No. 1,444,142, there is disclosed electronic data processing apparatus comprising:

40 keyboard input means for entering lines of one or more alphameric characters each into said electronic data processing apparatus;

45 first memory means for storing lines of one or more alphameric characters each;

50 second memory means, coupled to said keyboard input means and first memory means, adapted to store a single line of one or more alphameric characters entered into said electronic data processing apparatus from said keyboard input means or recalled from said first memory means; and

55 display means, coupled to said second memory means, adapted to visually display the single line of one or more alphameric characters stored in said second memory means;

60 said display means including means for visually displaying a cursor for

designating any character position of the displayed line of one or more alphameric characters;

said keyboard input means including cursor control means for controlling the position of said cursor; and

said electronic data processing apparatus including logic means, coupled to said keyboard input means, second memory means, and display means, for positioning said cursor to designate any character position of the displayed line of one or more alphameric characters in response to actuation of said cursor control means.

In the complete specification of our copending U.K. patent application No. 3346/76 (Serial No. 1,444,143), there is disclosed electronic data processing apparatus comprising:

keyboard input means, including a plurality of numeric keys, for entering lines of one or more alphameric characters each into said electronic data processing apparatus;

first memory means for storing lines of one or more alphameric characters each;

second memory means, coupled to said keyboard input means and first memory means, for storing a line of one or more alphameric characters entered into said electronic data processing apparatus from said keyboard input means or recalled from said first memory means; and

display means, coupled to said second memory means, for visually displaying the contents thereof;

said keyboard input means including a memory recall control key for recalling a designated line of one or more alphameric characters from said first memory means to said second memory means;

each line of one or more alphameric characters stored in said first memory means being associated with a separate line number; and

said electronic data processing apparatus including logic means, coupled to said keyboard input means and first and second memory means, for transferring a line of one or more alphameric characters from said first memory means to said second memory means in response to sequential actuation of said memory recall control key and one or more of said numeric keys designating the line number associated with the line of one or more alphameric characters to be transferred.

In the complete specification of our copending U.K. patent application No. 3347/76 (Serial No. 1,444,144), there is disclosed electronic data processing apparatus comprising:

keyboard input means, including a plurality of alphameric keys, for entering lines of one or more alphameric characters each into said electronic data processing apparatus;

memory means for storing lines of one or more alphameric characters each, every stored lines of one or more alphameric characters being associated with a separate line number;

buffer storage means, coupled to said keyboard input means and memory means, for storing a line number and for storing a line of one or more alphameric characters entered into said electronic data processing apparatus; and

display means, coupled to said buffer storage means, for visually displaying the contents thereof;

said keyboard input means including a control key for designating an automatic line numbering mode; and

said electronic data processing apparatus further including a line number counter for storing a current line number, temporary storage means for storing a line number increment and logic means coupled to said keyboard input means and buffer storage means, said logic means being responsive to actuation of said control key, followed by actuation of one or more alphameric keys designating both a starting line number and a line number increment, for storing the starting line number in both said buffer storage means and said line number counter and for storing the line number increment in said temporary storage means, said logic means being further responsive to completion of each entry of a line of one or more alphameric characters into the electronic data processing apparatus for combining the contents of said line number counter and said temporary storage means and for storing the result in both said line number counter and said buffer storage means.

In the complete specification of our copending application No. 3348/76 (Serial No. 1,444,145), there is disclosed electronic data processing apparatus comprising:

keyboard input means for entering one or more lines of one or more alphameric characters each into said electronic data processing apparatus;

keyboard buffer means, coupled to said keyboard input means, for storing a line of one or more alphameric characters entered into said electronic data processing apparatus; and

read-write memory means, coupled to said keyboard input means and keyboard buffer means, for storing one or more lines of more than three alphameric characters each entered into said electronic data processing apparatus;

said keyboard input means including a plurality of keys, each representative of a single alphabetic or numeric character, for entering a representation of every alphabetic character of a selected alphabet and every numeric character from zero to nine into said electronic data processing apparatus;

said electronic data processing apparatus including read-only memory means storing routines and subroutines for recognizing particular strings of alphameric characters entered into said electronic data processing apparatus as being commands; and

said electronic data processing apparatus including processing means, coupled to said keyboard input means, read-write memory means, and read-only memory means, for selectively performing the routines and subroutines stored in said read-only memory means and for executing the commands recognized thereby.

In the complete specification of our copending U.K. patent application No. 03349/76 (Serial No. 1,444,146), there is disclosed an electronic desk-top calculator comprising:

keyboard input means with a plurality of alphameric keys for entering BASIC language commands into the calculator and with an execute key for initiating execution of BASIC language command entered into the calculator;

storage means, coupled to said keyboard input means, for storing BASIC language commands entered into the calculator; and

processing means, coupled to said keyboard input means and storage means, for immediately executing a BASIC language command in response to entry of that BASIC language command into the calculator followed by actuation of said execute key.

In the complete specification of our copending application No. 3350/76 (Serial No. 1,444,147), there is disclosed electronic data processing apparatus comprising: keyboard input means for entering alphameric information into the apparatus; X—Y plotter means including a pen for graphically plotting selected alphameric information; memory means, coupled to said keyboard input means, for storing specified X and Y offset values; and processing means coupled to said keyboard input means, X—Y plotter means, and memory means, said processing means being responsive to an offset command entered into the apparatus for storing specified X and Y offset values into said memory means, said processing means being further responsive to subsequent plot commands, including specified X and Y coordinates, for summing each specified set of X and Y coordinates with the X and Y offset values stored in said memory means and for enabling plotting of the coordinates resulting from such summation.

In the complete specification of our copending application No. 3351/76 (Serial No. 1,444,148) there is disclosed electronic data processing apparatus comprising: keyboard input means with a plurality of keys for entering alphameric information into the data processing apparatus;

a modem for transmitting and receiving alphameric information over telephone lines; and

logic means coupled to said keyboard input means and modem, said logic means being responsive to entry of a terminal control command for selecting one or a plurality of baud rates at which alphameric information is to be transmitted from or received by the data processing apparatus.

In the complete specification of our copending application No. 3352/76 (Serial No. 1,444,149) there is disclosed an electronic calculator comprising:

keyboard input means for entering statements of one or more alphameric characters each into the calculator;

memory means, coupled to said keyboard input means, for storing statements entered into the calculator;

processing means, coupled to said keyboard input means and said memory means, for executing statements entered into the calculator or stored in said memory means;

magnetic tape reading and recording means for reading information from and recording information onto an external tape; and said processing means further including:

logic means coupled to said keyboard input means, memory means, and magnetic tape reading and recording means, said logic means being responsive to execution of a mark statement, entered from said keyboard input means or stored in said memory means, for electrically partitioning the external magnetic tape into one or more files prior to statements of one or more alphanumeric characters being transferred to the external magnetic tape.

The illustrated preferred embodiment of this invention is an electronic calculator employing a keyboard input unit, a magnetic tape cassette reading and recording unit, a solid state output display unit, an optional external output printer unit, an input-output control unit, a memory unit, and a central processing unit to provide an adaptable programmable calculator having manual operating, automatic operating, program entering, magnetic tape reading magnetic tape recording, and alphanumeric display and print modes. The keyboard input unit includes a group of data keys for entering numeric data into the calculator, a group of control keys for controlling the various modes and operations of the calculator and the format of the output display, a group of alphanumeric keys arranged as a typewriter keyboard for entering statements, and group of user-definable keys. All of the data and alphanumeric keys and some of the control keys may also be employed for programming the calculator.

The magnetic tape cassette reading and recording unit includes a reading and recording head, a drive mechanism for driving a magnetic tape past the reading and recording head, and reading and recording drive circuits coupled to the reading and recording head for bidirectionally transferring information between the magnetic tape and the calculator as determined by keyboard commands or commands which are part of a stored program.

The input-output control unit includes a sixteen-bit universal shift register serving as an input-output register into which information may be transferred serially from the central processing unit or in parallel from the keyboard input and magnetic tape cassette reading and recording units and from which information may be transferred serially to the central processing unit or in parallel to the solid state output display, magnetic tape cassette reading and recording, and output printer units. The input-output control unit preferably also includes control logic responsive to the central processing unit for controlling the transfer of information between these units. The input-output control unit may also be employed to perform the same functions between the central processing unit and peripheral units including, for example, an external printing unit, a digitizer, a marked card reader, an X-Y plotter, an external magnetic tape unit, a disc, a typewriter, and a modem. A plurality of peripheral units may be connected at the same time to the input-output control unit by simply plugging interface modules associated with the selected peripheral units into receptacles provided therefor in a rear panel of the calculator housing.

The memory unit includes a modular random-access read-write memory having a dedicated system area and a separate user area for storing program statements and/or data. The user portion of the read-write memory may be expanded without increasing the overall dimensions of the calculator by the addition of a program storage module. Additional read-write memory made available to the user is automatically accommodated by the calculator, and the user is automatically informed when the storage capacity of the read-write memory has been exceeded.

The memory unit also includes a modular read-only memory in which routines and subroutines of assembly language instructions for performing the various functions of the calculator are stored. These routines and subroutines of the read-only memory may be expanded and adapted by the user to perform additional functions oriented toward the specific needs of the user. This is accomplished by simply plugging additional read-only memory modules into receptacles provided therefor in a side panel of the calculator housing. Added read-only memory modules are automatically accommodated by the calculator and are accessed by the calculator through a series of mnemonic tables. These tables contain mnemonics which are additions to the calculator's programming language.

Plug-in read-only memory modules include, for example, a matrix module, a string variables module, a plotter module, an extended input-output module, and a

terminal module. The matrix module makes available to the user standard BASIC language matrix functions plus an additional function which returns the determinant of a previously defined square matrix. The string variables module makes available to the user standard BASIC language string variables operations. The plotter module enables the user to conveniently plot and label on an external X-Y plotter. The extended input-output module allows the calculator to be used with a wide variety of peripheral input-output units. The terminal module facilitates interfacing the calculator with a modem for communicating, for example, with remotely located time-sharing computer systems. It further allows free text editing and storage.

The memory unit further includes a pair of recirculating sixteen-bit serial shift registers. One of these registers serves as a memory address register for serially receiving information from an arithmetic-logic unit included in the central processing unit, for parallel addressing any memory location designated by the received information back to the arithmetic-logic unit. The other of these registers serves as a memory access register for serially receiving information from the arithmetic-logic unit, for writing information in parallel into any addressed memory location, for reading information in parallel from any addressed memory location, and for serially transferring information to the arithmetic-logic unit. It also serves as a four-bit parallel shift register for transferring four bits of binary-coded-decimal information in parallel to the arithmetic-logic unit.

The central processing unit includes four recirculating sixteen-bit serial shift registers, a four-bit serial shift register, the arithmetic-logic unit, a programmable clock, and a microprocessor. Two of these sixteen-bit serial shift registers serve as accumulator registers for serially receiving information from and serially transferring information to the arithmetic-logic unit. The accumulator register employed is designated by a control flip-flop. One of the accumulator registers also serves as a four-bit parallel shift register for receiving four bits of binary-coded-decimal information in parallel from and transferring four bits of such information in parallel to the arithmetic-logic unit. The two remaining sixteen-bit serial shift registers serve as a program counter register and a qualifier register, respectively. They are also employed for serially receiving information from and serially transferring information to the arithmetic-logic unit. The four-bit serial shift register serves as an extend register for serially receiving information from either the memory access register or the arithmetic-logic unit and for serially transferring information to the arithmetic-logic unit.

The arithmetic-logic unit is employed for performing one-bit serial binary arithmetic, four-bit parallel binary-coded-decimal arithmetic, and logic operations. It may also be controlled by the microprocessor to perform bidirectional direct and indirect arithmetic between any of a plurality of the working registers and any of the registers of the read-write memory.

The programmable clock is employed to supply a variable number of shift clock pulses to the arithmetic-logic unit and to the serial shift registers of the input-output, memory, and central processing units. It is also employed to supply clock control signals to the input-output control logic and to the microprocessor.

The microprocessor includes a read-only memory in which a plurality of microinstructions and codes are stored. These microinstructions and codes are employed to perform the basic instructions of the calculator. They include a plurality of coded and non-coded microinstructions for transferring control to the input-output control logic, for controlling the addressing and accessing of the memory unit, and for controlling the operation of the two accumulator registers, the program counter register, the extend register and the arithmetic-logic unit. They also include a plurality of clock codes for controlling the operation of the programmable clock, a plurality of qualifier selection codes for selecting qualifiers and serving as primary address codes for addressing the read-only memory of the microprocessor, and a plurality of secondary address codes for addressing the read-only memory of the microprocessor. In response to a control signal from a power supply provided for the calculator, control signals for the programmable clock, and qualifier control signals from the central processing and input-output control units, the microprocessor issues the microinstructions and codes stored in the read-only memory of the microprocessor as required to process either binary or binary-coded-decimal information entered into or stored in the calculator.

In the keyboard mode, the calculator is controlled by keycodes sequentially entered into the calculator from the keyboard input unit by the user. The solid state output display unit displays either the alphanumeric representation of the

keys as they are depressed or a numeric representation of output data or alphanumeric user instructions or program results. An external output printer unit may be controlled by the user to selectively print a numeric representation of any numeric data entered into the calculator from the keyboard input unit, a numeric representation of any result calculated by the calculator, or a program listing on a line-by-line basis of the statements entered.

When the calculator is in the keyboard mode, it may also be operated in a print-all printing mode. The output printer unit then prints out each program line as it is entered by the user.

In the program running mode, the calculator is controlled by automatically obtaining an internal representation of the program statements stored in the user storage section of the read-write memory. During automatic operation of the calculator, data may be obtained from the memory unit as designated by the program, from the keyboard input unit while the operation of the calculator is stopped for data either by the program or by the user, or from the magnetic tape cassette unit as designated by the program.

When the calculator is in the program running mode, the user may also selectively employ a trace mode to check the execution of the program line-by-line in order to determine whether the program, as entered into the calculator, does in fact carry out the desired sequence of statements.

In the program entering mode, statements are sequentially entered by the user into the calculator from the keyboard input unit and are translated into an internal stored format which consists of a series of operation codes and operand names and are thereafter stored as statements of a program in the user storage section of the read-write memory.

The magnetic tape cassette reading and recording unit may be employed by the user to separately load either data, BASIC language programs, assembly language programs, or sets of user-definable key definitions into the calculator from an external magnetic tape cassette.

The magnetic tape cassette reading and recording unit may also be employed by the user to separately record either data, BASIC language programs, or sets or user-definable key definitions stored in the user section of the read-write memory onto an external magnetic tape cassette. Programs, or portions thereof, may be coded by the user as being secure when they are recorded onto an external magnetic tape cassette. The calculator detects such programs when they are reloaded into the calculator and prevents the user from re-recording them or obtaining any listing or other indication of the individual program steps contained in the secured portions of such programs.

There now follows a detailed description which is to be read with reference to the accompanying drawings of a calculator according to the present invention; it is to be clearly understood that this calculator has been selected for description to illustrate the present invention by way of example and not by way of limitation.

In the drawings:—

Figure 1 is a front perspective view of an adaptable programmable calculator according to the invention;

Figure 2 is a rear perspective view of the adaptable programmable calculator of Figure 1;

Figures 3A—B are a simplified block diagram of the adaptable programmable calculator of Figures 1 and 2;

Figures 4A—F are a memory map of the memory unit employed in the adaptable programmable calculator of Figures 1 to 3;

Figure 4' is a diagram showing the arrangement of Figures 4A—F;

Figures 5A—B are a detailed memory map of the system read-write section of memory as shown in Figure 4A;

Figure 5' is a diagram showing the arrangement of Figures 5A—B.

Figure 6 is a detailed memory map of the user read/write section of memory as shown in Figure 4F.

Figures 7A—C are a simplified operational logic flow chart illustrating the operation of the microprocessor employed in the central processing unit of Figures 3A—B.

Figure 7' is a diagram showing the arrangement of Figures 7A—C.

Figure 8 is a plan view of the keyboard input unit employed in the adaptable programmable calculator of Figures 1—3B.

Figure 9 is an overall firmware block diagram for the adaptable programmable calculator.

Figures 10A—C are flow charts of floating point add and subtract routines selectable by the execution monitor of Figure 9.

Figure 11 is a flow chart of a floating point multiply routine selectable by the execution monitor of Figure 9.

Figures 12A—B are a flow chart of a floating point divide routine selectable by the execution monitor of Figure 9.

Figures 13A—C are a flow chart of a floating point square root routine selectable by the execution monitor of Figure 9.

Figure 14 is a flow chart of a store routine selectable by the execution monitor of Figure 9.

Figure 15 is a flow chart of a rounding routine employed in connection with several of the routines selectable by the execution monitor of Figure 9.

Figures 16A—B are a flow chart of a tangent X routine selectable by the execution monitor of Figure 9.

Figures 17A—B are a flow chart of an arctangent X routine selectable by the execution monitor of Figure 9.

Figures 18A—B are a flow chart of an e^x routine selectable by the execution monitor of Figure 9.

Figure 19 is a flow chart of a natural logarithm X routine selectable by the execution monitor of Figure 9.

Figure 20 is a flow chart of a subroutine employed by the tangent X and e^x routines of Figures 16A—B and 18A—B, respectively.

Figures 21A—B are a flow chart of a subroutine employed by the tangent X and arctangent X routines of Figures 16A—B and 17A—B, respectively.

Figures 22A—B are a flow chart of a subroutine employed by the e^x and natural logarithm X routines of Figures 18A—B and 19, respectively.

Figure 23 is a flow chart of a subroutine employed by the arctangent X and natural logarithm X routines of Figures 18A—B and 19, respectively.

Figure 24 is a flow chart of sine and cosine routines selectable by the execution monitor of Figure 9.

Figure 25 is a flow chart of an X^x power routine selectable by the execution monitor of Figure 9.

Figure 26 is a flow chart of a logarithm to the base ten routine selectable by the execution monitor of Figure 9.

Figure 27 is a block diagram of the microprocessor of Figures 3A—B.

Figures 28A—D are a detailed schematic diagram of the microprocessor of Figures 3A—B and 27.

Figure 28' is a diagram showing the arrangement of Figures 28A—D.

Figures 29A—J are detailed flow charts illustrating the operation of the microprocessor of Figures 3A—B, 27, and 28A—D.

Figures 29' and 29'' are diagrams showing the arrangement of Figures 29A—H.

Figure 30 is a block diagram of the programmable clock of Figures 3A—B.

Figures 31A—C are a detailed schematic diagram of the programmable clock of Figures 3A—B and 30 and of a portion of the input-output control unit of Figures 3A—B.

Figure 31' is a diagram showing the arrangement of Figures 31A—C.

Figure 32 is a waveform diagram illustrating the operation of the programmable clock of Figures 3A—B, 30, and 31A—C.

Figures 33A—D are a detailed schematic diagram of the shift register and arithmetic logic units of Figures 3A—B.

Figure 33' is a diagram showing the arrangement of Figures 33A—D.

Figure 34 is a block diagram of the arithmetic logic unit of Figures 3A—B.

Figure 35 is a block diagram of the memory unit of Figures 3A—B.

Figures 36A—B are a schematic diagram of the read-write memory of Figures 3A—B, 4A—F, and 35.

Figure 36' is a diagram showing the arrangement of Figures 36A—B.

Figures 37A—B are a schematic diagram of the optional add-on read-write memory of Figures 3A—B, 4A—F, and 35.

Figure 37' is a diagram showing the arrangement of Figures 37A—B.

Figure 38 is a schematic diagram of the basic read-only memory of Figures 3A—B, 4A—F, and 35.

Figure 39 is a schematic diagram of the optional add-on read-only memory modules of Figure 35 that may be plugged into the calculator to increase the number of functions available to the user.

Figure 40 is a detailed schematic diagram of the buffer circuitry associated with the read-only memory modules of Figures 39.

Figure 41 is a block diagram of one of the read-only memory chips of Figures 38 and 39.

Figures 42A—D are a schematic diagram of one of the read-only memory chips of Figures 38 and 39.

Figure 42' is a diagram showing the arrangement of Figures 42A—D.

Figure 43 is a memory map of the memory unit of Figures 3A—B and 4A—F illustrating how it is partitioned into the read-only and read-write memory chips of Figures 36A—B through 42A—D.

Figure 44 is a flow chart illustrating how the row members of the lists stored in the read-only memory chips are computed.

Figure 45 is a table of bit numbers and actual bits used in connection with the flow chart of Figure 44.

Figures 46A—B are a detailed schematic diagram of the memory address register of Figures 3A—B and 35 with its associated control circuitry.

Figure 46' is a diagram showing the arrangement of Figures 46A—B.

Figure 47A—B are a waveform diagram and state sequence charts illustrating the operation of the control circuitry of Figures 46A—B.

Figure 47' is a diagram showing the arrangement of Figures 47A—B.

Figures 48A—B are a detailed schematic diagram of the memory access register of Figures 3A—B and 35.

Figure 48' is a diagram showing the arrangement of Figures 48A—B.

Figures 49A—D are a detailed schematic diagram of the input-output register and gating control circuits employed in the input-output control unit of Figures 3A—B.

Figure 49' is a diagram showing the arrangement of Figures 49A—D.

Figure 50 is a schematic diagram of the source and relationship of the input-output party lines connected to the peripheral interface module receiving receptacles of Figure 20.

Figure 51 is a waveform diagram illustrating the operation of the control section of the input-output control unit of Figures 3A—B and 32.

Figure 52 is a flow chart illustrating the operation of the control section of the input-output control unit of Figures 3A—B and 31A—C.

Figure 53 is a diagram showing how a one-of-ten decoder is employed to address peripheral input-output units to the calculator through the input-output control unit of Figures 3A—B.

Figure 54 is a waveform diagram of some of the input signals employed by the input-output control unit and associated interface modules of Figures 3A—B.

Figure 55 is a waveform diagram of some of the output signals employed by the input-output control unit and associated interface modules of Figures 3A—B.

Figure 56 is a waveform diagram of some of the high speed input signals employed by the input-output control unit and associated interface modules of Figures 3A—B.

Figure 57 is a waveform diagram of some of the high speed output signals employed by the input-output control unit and associated interface modules of Figures 3A—B.

Figure 58 is a waveform diagram illustrating the operation of the interrupt mode of operation of the input-output control unit of Figures 3A—B.

Figure 59 is a schematic diagram of logic that may be used to interface an X—Y plotter to the input-output control unit of Figures 3A—B.

Figure 60 is a schematic diagram of logic that may be used to interface a printing unit to the input-output control unit of Figures 3A—B.

Figure 61 is a schematic diagram of logic that may be used to interface a modem to the input-output control unit of Figures 3A—B.

Figure 62 is a schematic diagram of logic that may be used to transfer any eight-bit code into or out of the input-output control unit of Figures 3A—B.

Figure 63 A—C are a detailed schematic diagram of the keyboard input unit employed in the adaptable programmable calculator of Figures 1—3B.

Figure 63' is a diagram showing the arrangement of Figures 63A—C.

Figure 64 is a block diagram of the magnetic tape cassette reading and recording unit employed in the calculator of Figures 1—3B.

Figure 65 is a detailed schematic diagram of the interface block of Figure 64.

Figures 66A—B are a detailed schematic diagram of the control logic block of Figure 64.

Figure 66' is a diagram showing the arrangement of Figures 66A—B.
Figures 67A—B are a detailed schematic diagram of the read-write block of Figure 64.

Figure 67' is a diagram showing the arrangement of Figures 67A—B.
Figures 68A—B are a detailed schematic diagram of the motor control block of Figure 64.

Figure 68' is a diagram showing the arrangement of Figures 68A—B.
Figure 69 is a detailed schematic diagram of the interconnect block of Figure 64.

Figure 70 is a detailed schematic diagram of the head driver and preamp blocks of Figure 64.

Figure 71 is a block diagram illustrating how the magnetic tape cassette reading and recording unit of Figures 64—70 interacts with the calculator of Figures 1—3B.

Figures 72A—F are a detailed schematic diagram of the output display unit employed in the adaptable programmable calculator of Figures 1—3B.

Figure 72' is a diagram showing the arrangement of Figures 72A—F.
Figures 73A—B are a detailed schematic diagram of the control logic circuit associated with the output display unit of Figures 72A—F.

Figure 73' is a diagram showing the arrangement of Figures 73A—B.
Figure 74 is a block diagram of the power supply system employed in the adaptable programmable calculator of Figures 1—3B.

Figures 75A—B are a detailed schematic diagram of the power supply system of Figure 74.

Figure 75' is a diagram showing the arrangement of Figures 75A—B.
Figures 76A—B are a block diagram of an interface module that may be employed to interface a typewriter to the adaptable programmable calculator of Figures 1—3B.

Figure 76' is a diagram showing the arrangement of Figures 76A—B.
Figures 77A—C are a flow chart of an input-output routine performed when a typewriter is employed with the programmable calculator of Figures 1—3B.

Figure 77' is a diagram showing the arrangement of Figures 77A—C.
Figures 78A—D are a detailed schematic diagram of the control logic block of Figures 76A—B.

Figure 78' is a diagram showing the arrangement of Figures 78A—D.
Figures 79A—B are a detailed schematic diagram of the power gates of Figures 76A—B.

Figure 79' is a diagram showing the arrangement of Figures 79A—B.
Figures 80A—B are a simplified logic diagram showing state qualifiers and instructions relating to the flow chart of Figures 77A—C.

Figure 81 is a detailed schematic diagram of the ROM, data latch, and compare circuitry of Figures 76A—B.

Figure 82 is a detailed schematic diagram of a power supply that may be employed to power the typewriter interface circuitry of Figures 78A—D, 79A—B, and 81.

Figure 83 is a flow chart of the turn-on routine which is a portion of the system monitor of Figure 4B.

Figures 84A—D are a flow chart of a routine comprising another portion of the system monitor of Figure 4B.

Figure 85 is a flow chart of the system table scan routine of Figure 4D.

Figure 86 is a flow chart of a subroutine called by the routine of Figure 85.
Figure 87 is a flow chart of the table search routine of Figure 4A.

Figures 88A—G are a flow chart of the keyboard input routine and special keyboard functions routines of Figure 4A.

Figures 89A—B are flow charts of subroutines called by the routines of Figures 88A—G.

Figure 90 is a flow chart of the clear subroutine called by various ones of the routines of Figures 4A—F.

Figure 91 is a flow chart of a subroutine called by the routine of Figure 88F.

Figure 92 is a flow chart of a subroutine called by the keyboard input routine of Figures 4A and 88A—G.

Figure 93 is a flow chart of a keyboard driver subroutine called by the subroutine of Figure 92.

Figures 94A—F are flow charts of some of the general system subroutines of Figure 4A.

	Figure 95 is a flow chart of the error routine of Figure 4B.	
	Figures 96A—D are flow charts of the program memory manager routines of Figure 4B.	
5	Figure 97 is a flow chart of another of the general system subroutines of Figure 4A which is also called by the routines of Figures 96A—D.	5
	Figure 98 is a flow chart of another of the general system subroutines of Figure 4A.	
	Figures 99A—F are flow charts of the user-definable key routines of Figure 4A.	
10	Figures 100A—D are flow charts of the execution monitor and keyboard execution control blocks of Figure 4C.	10
	Figure 101 is a flow chart of the load execution routine of Figure 4D.	
	Figure 102 is a flow chart of the store execution routine of Figure 4D.	
	Figure 103 is a flow chart of the merge execution routine of Figure 4D.	
15	Figure 104 is a flow chart of the link execution routine of Figure 4D.	15
	Figure 105 is a flow chart of the find execution routine of Figure 4D.	
	Figures 106A—D are flow charts of subroutines called by the execution routines of Figures 101—104.	
20	Figure 107 is a flow chart of the load data execution routine of Figure 4D.	20
	Figure 108 is a flow chart of the store data execution routine of Figure 4D.	
	Figure 109 is a flow chart of a subroutine called by the routines of Figures 107 and 108.	
	Figure 110 is a flow chart of the load key execution routine of Figure 4D.	
	Figure 111 is a flow chart of the store key execution routine of Figure 4D.	
25	Figure 112 is a flow chart of the load bin execution routine of Figure 4D.	25
	Figure 113 is a flow chart of the mark execution routine of Figure 4D.	
	Figure 114 is a flow chart of a subroutine called by the routine of Figure 113.	
	Figure 115A is a flow chart of the list execution routine of Figure 4D.	
30	Figure 115B is a flow chart of a subroutine called by the routine of Figure 115A.	30
	Figure 116 is a flow chart of the secure routine of Figure 4D.	
	Figures 117A—B are a flow chart of the interrupt routine of Figure 4D.	
35	Figures 118A—B are a flow chart of a routine for performing an OFFSET command selectable when the plotter plug-in read-only memory module is employed with the calculator.	35
	Figure 119 is a flow chart of a routine for performing an IPLOT command selectable when the plotter plug-in read-only memory module is employed with the calculator.	
40	Figure 120 is a flow chart of a routine for performing a LABEL command selectable when the plotter plug-in read-only memory module is employed with the calculator.	40
	Figure 121 is a flow chart of a routine for performing a LETTER command selectable when the plotter plug-in read-only memory module is employed with the calculator.	
45	Figure 122 is a flow chart of a routine for performing a CPLOT command selectable when the plotter plug-in read-only memory module is employed with the calculator.	45
	Figure 123 is a flow chart of a subroutine employed by the routine of Figure 122.	
50	Figure 124 is a flow chart of a routine for performing XAXIS and YAXIS commands selectable when the plotter plug-in read-only memory module is employed with the calculator.	50
	Figures 125A—C are flow charts of subroutines called by the routine of Figure 124.	
55	Figures 126A—C are flow charts of subroutines called by the routines of Figures 118—124.	55
	Figure 127 is a flow chart of a primary entry routine employed when the terminal read-only memory module is plugged into the calculator.	
60	Figures 128A—W are a flow chart of a keyboard input and editing routine employed when the terminal read-only memory module is plugged into the calculator.	60
	Figures 129A—B are a flow chart of a modem interrupt service routine employed when the terminal read-only memory module is plugged into the calculator.	

Figure 130 is a flow chart of a secondary entry routine employed when the terminal read-only memory module is plugged into the calculator.

Figure 131 is a flow chart of an output statement routine employed when the extended input-output read-only memory module is plugged into the calculator.

Figures 132A—D are flow charts of subroutines called by the routine of Figure 131.

Figure 133 is a flow chart of a BIN command selectable when the extended input-output plug-in read-only memory module is employed with the calculator.

Figure 134 is a flow chart of a CHAR command selectable when the extended input-output plug-in read-only memory module is employed with the calculator.

Figure 135 is a flow chart of a STAT command selectable when the extended input-output plug-in read-only memory module is employed with the calculator.

Figure 136 is a flow chart of binary ROTATE, AND, and OR commands selectable when the extended input-output plug-in read-only memory module is employed with the calculator.

Description of the Preferred Embodiment

GENERAL DESCRIPTION

Referring to Figures 1 and 2, there is shown an adaptable programmable calculator 10 including both a keyboard input unit 12 for entering information into and controlling the operation of the calculator and a magnetic tape cassette reading and recording unit 14 for recording information stored within the calculator onto one or more external tape cassettes 16 and for subsequently loading the information recorded on these and other similar magnetic tape cassettes back into the calculator. The calculator also includes a solid state output display unit 18 for displaying alphanumeric information stored within the calculator. All of these input and output units are mounted within a single calculator housing 24 adjacent to a curved front panel 26 thereof.

A plurality of peripheral input and output units including, for example, a line printer, a digitizer, a marked card reader, an X—Y plotter, a typewriter, a teletypewriter, an extended read-write memory unit, a magnetic disc reading and recording unit, and a modem for connecting the calculator via telephone lines to a remotely located computer, may be connected to the calculator at the same time by simply inserting interface modules 30 (Figure 3) associated with the selected peripheral units into any of four receptacles 32 provided therefor in a rear panel 34 of the calculator housing. As each interface module 30 is inserted into one of these receptacles, a spring-loaded door 38 at the entrance of the receptacle swings down allowing passage of the interface module. Once the interface module is fully inserted, a printed-circuit terminal board 40 contained within the interface module plugs into a mating edge connected mounted inside the calculator. If any of the selected peripheral units require AC line power, their power cords may be plugged into either one of two AC power outlets 42 provided therefor at the rear panel of the calculator housing 24.

Referring to the simplified block diagram shown in Figures 3A—B, it may be seen that the calculator also includes an input-output control unit 44 (hereinafter referred to as the I/O control unit) for controlling the transfer of information to and from the input and output units, a memory unit 46 for storing and manipulating information entered into the calculator and for storing routines and subroutines of basic instructions performed by the calculator, and a central processing unit 48 (hereinafter referred to as the CPU) for controlling the execution of the routines and subroutines of basic instructions stored in the memory unit as required to process information entered into or stored within the calculator. The calculator also includes a bus system comprising an S-bus 50, a T-bus 52, and an R-bus 54 for transferring information from the memory and I/O control units to the CPU, from the CPU to the memory and I/O control units, and between different portions of the CPU. It further comprises a power supply for supplying DC power to the calculator and peripheral units employed therewith and for issuing a control signal POP when power is supplied to the calculator.

The I/O control unit 44 includes an input-output register 56 (hereinafter referred to as the I/O register), associated I/O gating control circuitry 58, and input-output control logic 60 (hereinafter referred to as the I/O control). I/O register 56 comprises a universal sixteen-bit shift register into which information may be transferred either bit-serially from the CPU 48 via T-bus 52 or in parallel from keyboard input unit 12, magnetic tape cassette reading and recording unit 14, and peripheral input units 28 such as the marked card reader via twelve input party

lines 62. Information may be transferred from I/O register 56 either bit-serially to CPU 48 via S-bus 50 or in parallel to magnetic tape cassette reading and recording unit 14, solid state output display unit 18, output printer unit 20, and peripheral output units 28 such as the X—Y plotter or the typewriter via sixteen output party lines 64.

I/O gating control circuitry 58 includes control circuits for controlling the transfer of information into and out of I/O register 56 in response to selected I/O qualifier control signals from CPU 48 and selected I/O control instructions from I/O control 60. It also includes an interrupt control circuit 65, a peripheral control circuit 66, a printer control circuit 68, and a display control circuit 69 for variously controlling the input and output units and issuing control signals QFG and EBT to I/O control 60 via two output lines 71 and 72. These last mentioned control circuits variously perform their control functions in response to control signal POP from the power supply, I/O qualifier control signals from CPU 48, I/O control instructions from I/O control 60, and control signals from keyboard input unit 12. Interrupt control circuit 65 initiates the transfer of information into I/O register 56 from keyboard input unit 12 or interrupting peripheral input units 28 such as the marked card reader and issues a qualifier control signal QNR to CPU 48 via output lines 73. Peripheral control circuit 66 enables interface modules 30 plugged into the calculator to respond to information from I/O register 56, control associated peripheral units 28, transfer information to and/or receive information from associated peripheral units 28, and in some cases initiate the transfer of information to I/O register 56 from the interface modules themselves. Printer control circuit 68 and display control circuit 69 enable output display unit 18, and output printer unit 20, respectively, to respond to information from I/O register 56.

When a basic I/O instruction obtained from memory unit 46 is to be executed, CPU 48 transfers control to I/O control 60 by issuing a pair of I/O microinstructions PTR and XTR thereto. In response to these I/O microinstructions from CPU 48, control signal POP from the power supply, control signals QFG and EBT from I/O gating control circuitry 58, and I/O qualifier and clock control signals from CPU 48, I/O control 60 selectively issues one or more I/O control instructions to gating control circuitry 58 as required to execute the basic I/O instructions designated by CPU 48 and issues control signals, TTX, XTR, QRD, and SCB to CPU 48 via output lines 74—77. The I/O qualifier control signals issued to I/O control 60 and gating control circuitry 58 by CPU 48 are derived from the basic I/O instruction to be executed. Those qualifier control signals issued to I/O control 60 designate the specific I/O control instructions to be issued by I/O control 60, while those issued to gating control circuitry 58 designate selected control circuits to be employed in executing the basic I/O instruction.

Memory unit 46 includes a modular random-access read-write memory 78 (hereinafter referred to as the RWM), a modular read-only memory 80 (hereinafter referred to as the ROM), a memory address register 82 (hereinafter referred to as the M-register), a memory access register 84 (hereinafter referred to as the T-register), and control circuitry 85 for these memories and registers. The RWM 78 and ROM 80 comprise MOS-type semiconductor memories. As shown in the memory map of Figures 4A—F the basic RWM 78 contains a dedicated system storage section of 256 sixteen-bit words extending from address 1400 to address 1777 and a separate user program and/or data storage section of 1792 sixteen-bit words extending from address 40400 to address 43777. All addresses on the memory map are represented in octal form.

An optional 2048 sixteen-bit words of RWM may be made available to the user at address 44000 to address 47777. This is accomplished by removing a top panel 90 of the calculator housing shown in Figure 1, and inserting an additional printed circuit board containing the optional memory. The additional RWM is automatically accommodated by the calculator.

As shown in the more detailed memory map of Figures 5A—B, the RWM dedicated system storage section includes 52 words (addresses 1400—1463) containing information in the form of mnemonic variables which is employed by the firmware routines shown in Figure 9. A more detailed description of these mnemonic variables is given on page 21 of the calculator basic system firmware listing located hereinafter in this specification. Addresses 1466—1477 and 1701—1737 are used as temporary storage by the various routines shown in Figure 9. Addresses 1500—1550 comprise a 41-word buffer used to contain the input characters during syntax analysis. Addresses 1551—1621 comprise a 41-word

buffer used by the single line display refresh routine of Figure 9. These 41 words along with 42 additional words (addresses 1622—1673) are used as a syntax buffer during syntax analysis. Four of these words (addresses 1622—1625) are used as temporary registers by several of the statement execution routines of Figure 9. Eight words (addresses 1630—1637) are used as two temporary floating point number registers by the formula evaluation routines of Figure 9. Addresses 1640—1677 comprise 32 words which are used by the statement execution routines of Figure 9. Eight words (addresses 1744—1747 and 1754—1757) are employed as "AR1" and "AR2" four-word working registers for performing binary-coded-decimal arithmetic. An additional eight words (addresses 1740—1743 and 1750—1753) are employed as working data registers "X_e" and "Y_e" for implementation of the trigonometric functions. These sixteen words (addresses 1740—1757) are used as temporary storage registers by all the routines of Figure 9 except the statement execution and formula evaluation routines. A variable-length "system subroutine stack" (addresses 1760—1772) is employed for storing return addresses required by programs stored in ROM 80. Four words (addresses 1773—1776) are used to store the result of the latest keyboard computation. The last word in the system RWM (address 1777) is used to store a pointer indicating the next available location for the return address of the next subroutine call within the basic system. A complete assembly language description of the system RWM is included in pages 21—24 of the calculator basic system firmware listing.

As shown in the memory map of Figures 4A—F and the more detailed memory map of Figure 6, user program and/or data storage section of RWM 78 contains 1760 word available to the user (as user addresses 40440—43777) for storing programs and/or data, 20 words dedicated for use by the interrupt routine of Figures 9 and 117A—B, and 12 words available for use by plug-in read-only memory modules. An additional 2048 sixteen-bit words may be made available to the user (as user addresses 44000—47777).

Also, as shown in the memory map of Figures 4A—B, the basic ROM 80 contains 7680 sixteen-bit words extending from address 0000 to address 1377, from address 2000 to address 16777, and from address 40000 to address 40377. Routines and subroutines of basic instructions for performing the basic functions of the calculator and constants employed by these routines and subroutines are stored in these portions of ROM 80. An additional 8192 sixteen-bit words of ROM may also be added at addresses 20000—37777 in steps of 512 and 1,024 words. This is accomplished by simply inserting plug-in ROM modules 92 into receptacles provided therefor within the calculator which are accessible through a door in the left panel of the calculator housing as illustrated in Figure 1. As each plug-in ROM module 92 is inserted into one of these receptacles a printed circuit terminal board 96 contained within the plug-in ROM module plugs into a mating edge connector mounted inside the calculator. A handle pivotally mounted at the top end of each plug-in ROM module 92 facilitates removal of the plug-in ROM module once it has been fully inserted into one of the receptacles.

Routines and subroutines of basic instructions (and any needed constants) for enabling the calculator to perform many additional functions are stored in each plug-in ROM module 92. The user himself may therefore quickly and simply adapt the calculator to perform many additional functions oriented toward his specific needs by simply plugging ROM modules of his own choosing into the calculator. Added plug-in ROM modules are automatically accommodated by the calculator.

Referring again to Figures 3A—B, M-register 82 of the memory unit comprises a recirculating sixteen-bit serial shift register into which information may be transferred bit-serially from CPU 48 via T-bus 52 and out of which information may be transferred bit-serially to CPU 48 via S-bus 50. Information shifted into M-register 82 may be employed to address any word in RWM 78 or ROM 80 via fifteen output lines 106.

T-register 84 of the memory unit comprises a recirculating sixteen-bit serial shift register into which information may be transferred either bit-serially from CPU 48 via T-bus 52 or in parallel from any addressed word in RWM 78 and ROM 80 via sixteen parallel input lines 108. Information may be transferred from T-register 84 either bit-serially to CPU 48 via S-bus 50 or parallel to any addressed word in RWM 78 via sixteen parallel output lines 110. The four least significant bits of information contained in T-register 84 may comprise binary-coded-decimal information and may be transferred from the T-register in parallel to CPU 48 via three parallel output lines 112 taken with S-bus 50.

The control circuitry 85 of the memory unit controls these transfers of

information into and out of M-register 82 and T-register 84, controls the addressing and accessing of RWM 78 and ROM 80, and refreshes RWM 78. It performs these functions in response to memory microinstructions, memory clock pulses, and shift clock pulses from CPU 48.

CPU 48 includes a register unit 114, an arithmetic-logic unit 116 (hereinafter referred to as the ALU), a programmable clock 118, and a microprocessor 120. Register unit 114 comprises four recirculating sixteen-bit shift registers 122, 124, 126, and 128 and one four-bit shift register 130. Shift registers 122 and 124 serve as sixteen-bit serial accumulator registers (hereinafter referred to as the A-register and the B-register, respectively) into which information may be transferred bit-serially from ALU 116 via T-bus 52 and out of which information may be transferred bit-serially to ALU 116 via R-bus 54. The four least significant bit positions of A-register 122 also serve as a four-bit parallel accumulator register into which four bits of binary-coded-decimal information may be transferred in parallel from ALU 116 via four parallel input lines 132 and out of which four bits of binary-coded-decimal information may also be transferred in parallel to ALU 116 via three parallel output lines 134 taken with R-bus 54.

Shift register 126 serves as a sixteen-bit system program counter (hereinafter referred to as the P-register) into which information may be transferred bit-serially from ALU 116 via T-bus 52 and out of which information may be transferred bit-serially to ALU 116 via R-bus 54. Information contained in the least significant bit position of P-register 126 may also be transferred as a qualifier control signal QPO to microprocessor 120 via output line 135.

Shift register 128 serves as a sixteen-bit qualifier register (hereinafter referred to as the Q-register) into which information may be transferred bit-serially from ALU 116 via T-bus 52 and out of which information may be transferred bit-serially to ALU 116 via R-bus 54. Information contained in the five least significant bit positions of Q-register 128 is transferred to I/O gating control circuitry 58 as five one-bit I/O qualifier control signals Q00—Q04 via five parallel output lines 136, and information contained in the six next least significant bit positions of the Q-register is transferred to I/O control 60 as six one-bit I/O qualifier control signals Q05—Q10 via six parallel output lines 138. Similarly, information contained in the seven least significant, the ninth and eleventh least significant, and the most significant bit positions of Q-register 128 and information derived from the thirteenth, fourteenth, and fifteenth bit positions of the Q-register may be transferred to microprocessor 120 as eleven one-bit microprocessor qualifier control signals Q00—Q06, Q08, Q10, Q15, and QMR via eleven output lines 140. Information contained in the twelfth through the fifteenth least significant bit positions of Q-register 128 may be transferred to microprocessor 120 as a four-bit primary address code via four parallel output lines 142.

Shift register 130 serves as a four-bit serial extend register (hereinafter referred to as the E-register) into which information may be transferred bit-serially either from ALU 116 via T-bus 52 or from the least significant bit position of T-register 84 via input line 144. Information may also be transferred out of E-register 130 to ALU 116 via R-bus 54.

Register unit 114 also includes control circuitry 146 for controlling the transfer of parallel binary-coded-decimal information into and out of A-register 122 and the transfer of serial binary information into and out of A-register 122, B-register 124, P-register 126, Q-register 128, and E-register 130. This is accomplished in response to register microinstructions from microprocessor 120, control signals TTX and XTR from I/O control 60, and shift clock control pulses from programmable clock 118. Control circuitry 146 includes a flip-flop 148 (hereinafter referred to as the A/B flip-flop) for enabling the transfer of information into and out of either the A-register 122 or the B-register 124 as determined by the state of the A/B flip-flop. The state of A/B flip-flop 148 is initially determined by information Q11 transferred to the A/B flip-flop from the twelfth least significant bit position of Q-register 128 but may be subsequently complemented one or more times by microinstruction CAB from microprocessor 120.

ALU 116 may perform either one-bit serial binary arithmetic on data received from T-register 84 or M-register 82 via S-bus 50 and/or from any register of register unit 114 via R-bus 54 or four-bit parallel binary-coded-decimal arithmetic on data received from T-register 84 via output lines 112 taken with S-bus 50 and/or from A-register 122 via output lines 134 taken with R-bus 54. It may also perform logic operations on data received from memory unit 46 and/or register unit 114 via any

of these lines. The arithmetic and logic operations performed are designated by ALU microinstructions from microprocessor 120 and are carried out in response to these microinstructions, shift clock control pulses from programmable clock 118, and control signal SCB from I/O control 60. Information is also transferred from ALU 116 to A-register 122 via output lines 132 or to I/O register 56, M-register 82, T-register 84, or any register of register unit 114 via T-bus 52 in response to microinstructions and control signals applied to these registers. If a carry results while ALU 116 is performing either one-bit serial binary arithmetic or four-bit parallel binary-coded-decimal arithmetic, the ALU issues a corresponding qualifier control signal QBC or QDC to microprocessor 120 via one of two output lines 152 and 154.

Programmable clock 118 includes a crystal-controlled system clock 156, a clock decoder and generator 158, and a control gate 160. System clock 156 issues regularly recurring clock pulses to clock decoder and generator 158 via output line 162. In response to these regularly recurring clock pulses from system clock 156 and to four-bit clock codes from microprocessor 120, clock decoder and generator 158 issues trains of n shift clock pulses to ALU 116, M-register 82, T-register 84, and all of the registers of register unit 114 via output line 164. These trains of n shift clock pulses are employed for shifting a corresponding number of bits of serial information into or out of any of these registers or for shifting a carry bit in the ALU. The number n of pulses in each of these trains may vary from one to sixteen as determined by the number of bits of serial information required during each operation to be performed. In response to a control signal CCO from microprocessor 120, control gate 160 prevents any shift clock pulses from being applied to the ALU or any of these registers. Upon completion of each train of n shift clock pulses, clock decoder and generator 158 issues a ROM clock pulse to microprocessor 120 via output line 166 and an I/O clock pulse to I/O control 60 via output line 168. In response to the regularly recurring clock signal from system clock 56, clock decoder and generator 158 also issues correspondingly regularly recurring memory clock pulses to memory unit 46 via output line 170.

Microprocessor 120 selectively issues two I/O microinstructions to I/O control 60 via two output lines 172, six memory microinstructions to memory unit 46 via six output lines 174, thirteen register microinstructions to register unit 114 via thirteen output lines 176, and five ALU microinstructions to ALU 116 via five output lines 178. It also issues a four-bit clock code associated with each of these microinstructions to clock decoder 158 via four output lines 180. These microinstructions and associated clock codes are issued as determined by the control signal POP from the power supply, the eleven microprocessor qualifier control signals from Q-register 128, the four-bit primary address codes from Q-register 128, and the five microprocessor qualifier control signals from I/O control 60, interrupt control 65, ALU 116, and P-register 126.

As shown in the simplified flow chart of Figure 7, microprocessor 120 executes a hardware diagnostic routine (stored within the microprocessor itself) in response to the control signal POP. Upon completion of this diagnostic routine, ALU 116 issues the qualifier control signal QBC indicating whether or not the diagnostic routine was successful. Microprocessor 120 thereupon responds to this qualifier control signal by entering the basic machine operating loop and issuing microinstructions causing a sixteen-bit instruction stored in ROM 80 to be loaded into T-register 84 and transferred from there to Q-register 128. Microprocessor 120 thereupon sequentially responds to one or more additional qualifier control signals by issuing microinstructions and associated clock codes for executing the instructions then contained in Q-register 128 and causing another sixteen-bit instruction stored in ROM 80 to be loaded into T-register 84 and transferred from there to the Q-register. When an instruction requiring multiple branching is contained in Q-register 128, microprocessor 120 issues a pair of microinstructions UTR and XTR causing the microprocessor to respond to a four-bit primary address code from the Q-register by issuing additional microinstructions and associated clock codes for executing the instruction contained in the Q-register.

As illustrated by the basic machine operating loop shown in the flow chart of Figure 7, microprocessor 120 initially responds to the qualifier control signal QNR either by issuing microinstructions and associated clock codes for interrupting the basic machine operating loop and executing an I/O service routine or by issuing microinstructions and associated clock codes for loading A/B flip-flop 148 with the information Q11 contained in Q-register 128. The manner in which microprocessor 120 responds is determined by the condition of the qualifier control

85

signal QNR, which in turn indicates whether or not the basic machine operating loop should be interrupted.

5 Assuming the basic machine operating loop is not to be interrupted, microprocessor 120 loads the information Q11 into A/B flip-flop 148 and responds to the qualifier control signal QMR either by issuing microinstructions for transferring an address portion of the instruction contained in Q-register 128 from T-register 84 into M-register 82 or by responding to another qualifier control signal Q15. Again, the manner in which microprocessor 120 responds is determined by the condition of the qualifier control signal QMR, which in turn indicates whether or not the instruction contained in Q-register 128 is a memory reference instruction. 10

Assuming the instruction contained in Q-register 128 is a memory reference instruction, microprocessor 120 transfers the required address information into the M-register 82 and responds to qualifier control signal Q10 either by issuing microinstructions and associated clock codes to select the base page of the memory (i.e. page 0) or by issuing microinstructions and associated clock codes to select the current page of the memory (i.e. the page from which the instruction contained in Q-register 128 was obtained). In either case, the microprocessor then issues microinstructions as required to read data from the preset page of the memory at the address designated by the address information last transferred into M-register 82. Upon completion of this operation, microprocessor 120 responds to qualifier control signal Q15 by issuing additional microinstructions and associated clock codes to execute an indirect memory access operation if the condition of this qualifier control signal indicates that the address information contained in M-register 82 is indirect. 15 20

Assuming the address information contained in M-register 82 is direct (or upon completion of the indirect memory access operation), microprocessor 120 issues microinstructions and associated clock codes causing the microprocessor itself to respond to a four-bit primary address code from the Q-register. The microprocessor responds by issuing additional microinstructions and associated clock codes for executing whichever one of ten possible memory reference instructions is contained in Q-register 128 and designated by the four-bit primary address code. Following execution of the designated memory reference instruction, microprocessor 120 issues microinstructions and associated clock codes causing another sixteen-bit instruction stored in ROM 80 to be loaded into T-register 84 and transferred from there to Q-register 128, thereby beginning another cycle of the basic machine operating loop. 25 30 35

As illustrated by other possible paths of the basic machine operating loop shown in Figure 7, microprocessor 120 sequentially responds to other qualifier control signals when other types of instructions are contained in Q-register 128. For example, when an I/O instruction is contained in Q-register 128, microprocessor 120 sequentially responds to qualifier control signals QNR, QMR, Q15, Q10, and QRD by issuing microinstructions and associated clock codes to execute the I/O instruction. It should be noted that the microprocessor qualifier control signals not shown in the simplified flow chart of Figure 7 are variously contained within those flow chart blocks requiring decisions as will hereinafter become apparent. 40 45

The calculator firmware operational diagram of Figure 9 illustrates the basic components of the calculator firmware. These components comprise routines which reside in the calculator ROM 80 and serve to implement the definition of the calculator. Control information passing between routines is represented by solid lines on the drawing. 50

Referring to Figure 9, it is shown that the calculator hardware units are controlled by firmware routines contained in ROM 80. These units comprise an on-off power switch 182, an alphanumeric keyboard input unit 12, a display unit 18, and a magnetic tape cassette reading and recording unit 14. The firmware routines also control an external printer 20, external tape cassette reading and recording units 14, and various other external input-output devices 244. 55

Operation of the calculator is begun by placing the on-off switch in the "on" position, thus forcing the hardware internal to the calculator to execute the instruction located at address 0000 of ROM 80. This instruction directs control to the start-up routine 200, which is shown in the flowchart of Figure 83, and described in detail on page 53 of the basic system firmware listing. The purposes of this routine are to initialize RWM 78, set the stack pointer address at location 1777, set the keyboard execution numeric output format to float 9, initialize certain variables in the system RWM area for later use by other firmware routines, and 60 65

initialize the various read-write pointers to the user read-write memory area shown in detail in the memory map of Figure 6.

After completion of the start-up routine, control is passed to the keyboard monitor routine 202 shown in the flowchart of Figures 84A—D and detailed on pages 53—56 of the basic system firmware listing. This routine initializes certain variables in the system RWM 78 for use by the keyboard input routine 204. It also outputs the automatic line number if necessary. It then calls for an input record from the keyboard input routine 204. When the keyboard input routine returns with a record the keyboard monitor routine searches the mnemonic tables in the assembly language program area shown in Figure 6. It then searches the mnemonic tables in each of the plug-in read-only memory modules of Figure 4E and finally searches the mnemonic tables of the main system ROM 80. A complete assembly language listing of each of the tables in read-only memory is given in the firmware listings. The subroutines which do the search of the mnemonic tables are detailed in the flowcharts of Figures 85, 86, and 87. If a match is found between the characters of the input record and any of the mnemonic tables, the keyboard monitor branches through a jump table to the appropriate syntax routine 210 for syntax analysis if the mnemonic is a statement, or branches through a jump table for execution if the mnemonic is a system command. A separate syntax analysis routine is provided for each statement and a separate execution routine is provided for each system command. Syntax and execution routines for statements and commands on an optional read-only memory module are contained in the firmware of that module. The assembly language program area is handled in the same fashion as the plug-in read-only memory module. If no mnemonic is found, control is passed to the implied LET syntax routine.

The keyboard input routine 204 is detailed in the flowcharts of Figures 88A—G, 89A—B, and 90—93. It calls on the display refresh routine 206 to refresh the 32 character single line display 18 between key entries. The display refresh routine is detailed on page 35 of the firmware listing. When a key is entered through the alphanumeric keyboard 12, the interrupt circuitry causes the calculator to execute the instruction at address 00002. This instruction causes a jump to the interrupt routine 208, which is detailed in the flowchart of Figures 117A—B and pages 257 and 258 of the firmware listing. The interrupt routine 208 saves the keycode in a memory location of the system RWM 78 and returns. The keyboard input routine 204 reads this memory word and decides what operations need to be performed for that particular keycode. The shift bit is stripped from the keycode and stored as a flag in a temporary location in system RWM 78. If the key requires that a mnemonic name be displayed, the single line display buffer shown in Figures 5A—B is cleared and the mnemonic name is entered. If an editing function is required, a routine is called to perform the editing function. If a user-definable key f0—f9 has been given, the user-definable key routine 228 is called. If an alphanumeric key has been given, the shift flag in RWM 78 is tested, and, if the shift has been given, the keycode is converted to the code for the shifted key. Then the keycode is inserted into the single line display buffer shown in Figures 5A—B, either at the end of the line or at the cursor position, if the cursor is within the line.

The user-definable key routines 228 perform the special operations for keys f0—f9. They are detailed in the flowcharts of Figures 99A—F and pages 43—47 of the firmware listing.

The syntax routines 210 translate the characters of the input record into an internal format which is more easily handled by the execution routines. If a syntax error is encountered control is passed to the error routine 238 which outputs an error message. The error routine is shown in the flowchart of Figure 95 and the firmware listing at pages 58 and 59. If no error is found, control is passed to either the memory management routines 236, if the statement is to be stored in memory, or to the routine for initialization for keyboard execution if the statement is to be executed. The memory management routines are shown in the flowcharts of Figures 96A—D, 97, and 98 and pages 16 and 60—62 of the firmware listings. The routine 230 for initialization for keyboard execution serves to initialize the run time stacks shown in the memory map of Figure 6. This routine is detailed in the flowchart of Figure 100C and pages 109—110 of the firmware listing.

When the RUN command is given, or if the INIT key is actuated, control is passed to the pre-execution processing routines 232. These routines are detailed in pages 65—77 of the firmware listing. They serve to initialize the symbol table and non-common value table areas of the user read-write memory shown in Figure 6.

Control is next passed to the execution monitor 214, which is detailed in the flowchart of Figures 100A—B and pages 108—110 of the firmware listing. This routine initializes the run time stacks in the user read-write memory of Figure 6 and initiates execution of a stored program beginning at the line number given by the user. After each statement is executed control is returned to the execution monitor, which prints the line number of the next line if the program is being executed in the trace mode. Step, stop, or error conditions are checked and program execution is terminated if any of these conditions exist. If program execution is to be continued, the jump address of the execution routine for the next statement of the program is computed, and control is passed to that routine.

Several of the statement execution routines 240 require evaluation of arithmetic functions and expressions. This is done in the formula evaluation routines 242. Several of the statement execution routines 240 require input from or output to various external input-output devices 20 and 244. This is done by calling the standard output driver 224 or an optional special I/O driver 234. Several of the statement execution routines require input from or output to an internal or external tape cassette unit 14. This is done by calling the tape cassette drivers 226. The statement execution routines for the statements that communicate with the tape cassette units are detailed in the flowcharts of Figures 101—115 and pages 250—276 of the firmware listing.

The list routine 220 is used when listing stored programs on either the single line display 18 or an external ASCII output device 20. The list routine is detailed on pages 78—82 of the firmware listing. Its function is to translate the stored program from the internal stored format into a string of ASCII characters which can be printed or displayed. To print a line of translated characters, the list routine calls on the standard output driver 242 which is detailed in Figure 94E and pages 20 and 33 of the firmware listing.

Detailed assembly language information relating to all of the firmware routines and subroutines herein described may be obtained by referring to the memory map of Figures 4A—F and the basic system firmware listing located at a later point in this specification.

Communication with the routines in the various plug-in read-only memory modules is accomplished through a series of mnemonic tables and jump tables. The standard firmware, the cassette operating firmware, and each of the plug-in read-only memory modules all contain the following tables:

- (1) A statement mnemonic table
- (2) A statement syntax jump table
- (3) A statement execution jump table
- (4) A system command mnemonic table
- (5) A system command execution jump table
- (6) A function mnemonic table
- (7) A function execution jump table
- (8) A non-formula operator mnemonic table

All of these tables, with the exception of the statement execution jump table, may appear anywhere within a memory module. The last five words in each module are used by the table scan routines of Figures 85—87 to find the actual location of the tables. The last word of each module contains a unique operation code word for that particular module. The second from last word contains a relative address of the statement mnemonic table. The third from the last word contains a relative address of the system command mnemonic table. The fourth from the last word contains a relative address to the function mnemonic table. The fifth from the last word contains a relative address to the non-formula operator table. The jump tables for statement syntax, system command execution, and function execution are located directly above their respective mnemonic tables. The statement execution jump table is located directly above the fifth from the last word of each module. The complete set of tables for the standard firmware is shown in the firmware listings at pages 48—49, 103—104, 131, and 141 of the listings.

Each of the mnemonic tables consists of a string of seven-bit ASCII character and six-bit operation code characters packed two characters per sixteen-bit word. The eight bit of each character is used to indicate whether that character is ASCII or an operation code. A zero in the eighth bit indicates ASCII and a one indicates an operation code. The seventh bit of each operation code character is used to indicate whether that operation code is the last character in that table. The jump table address for each mnemonic is found by subtracting the operation code for

that mnemonic from the starting address of the associated mnemonic table. The internal stored format for program statements consists of a series of operation codes, operand codes, and other special codes. The first word of each statement contains the line number of that statement in binary format. The second word contains both the operation code for that particular statement mnemonic and also the length of the statement. The length information is used by various firmware routines to scan from one statement to the next. The third word contains the operation code for the table or optional read-only memory module, and it also contains the first operand code. The remainder of the statement is stored with one operation code and one operand code in each word.

Formula operation codes from the table on page 132 of the firmware listing and mnemonic operation codes are stored in a five-bit field, bits 10-14. The operand codes are stored in two five-bit fields. Bits 5-9 are used to store the operand name. The name consists of an ASCII letter, A-Z, with its sixth and seventh bits removed. For example, the ASCII code for A is 1000001 and the five-bit operand code for A is 00001. Bits 0-4 are used to store the operand type. Bit 15 is used as a special flag bit. When bit 15 is set, the operand field is interpreted differently than when it is not set. The following table shows the various operand types and the special codes.

37

OPERAND TYPE CODE	OPERAND TYPE (BIT 15=0)	MEANING IF BIT 15=1	
		<u>Full Precision Variables</u>	<u>Constant follows in next word</u>
00000	Simple variable		
00001	Array of 1 dimension		Fixed point decimal
00010	Array of 2 dimensions		Floating point decimal
00011	Array of unknown dimension		Binary integer
	<u>Split Precision Variables</u>		
00100	Simple variable		Binary line number
00101	Array of 1 dimension		
00110	Array of 2 dimensions		
	<u>Integer Precision Variables</u>		
01000	Simple variable		
01001	Array of 1 dimension		
01010	Array of 2 dimensions		
	<u>Full Precision Variables</u>		
	<u>Letter followed by digit</u>		
10000	0		
10001	1		
10010	2		
10011	3		
10100	4		
10101	5		
10110	6		
10111	7		
11000	8		
11001	9		
11110	String Variable		
11111	User defined function	{ Bits 5-9 contain the operation code of a function in ROM	

As an example, the internal stored format for the following statements is shown in the table below:

				10 DIM A[5]
				20 LET B=C+FND(E)
				30 GOTO 100
15	14-10	4-5	4-0	
0	00000	00000	01010	10
0	00010	00000	00111	DIM op-code -- Length 7
0	01010	00001	00001	Table op-code -- A dimension
1	10100	00000	00011	[-- integer follows
0	00000	00000	00101	5
0	00101	00000	00000	] -- null operand
0	00000	00000	10100	20
0	10110	00000	00111	LET op-code -- Length 7
0	01010	00010	00000	Table op-code -- B
0	00111	00011	00000	= -- C
0	01001	00100	11111	+ -- FND
0	10101	00101	00000	(-- E
0	00100	00000	00000	) -- null operand
0	00000	00000	11110	30
0	00110	00000	00100	GOTO op-code -- Length 4
1	01010	00000	00100	Table op-code -- integer follows
0	00000	00011	00100	100

KEYBOARD OPERATIONS

All operations performed by the calculator may be controlled or initiated by the keyboard input unit and/or by keycodes entered into the calculator from the keyboard input unit, the magnetic tape cassette reading and recording unit, or peripheral input units such as the marked card reader and stored as program steps in the program storage section of the RWM. An operational description of the keyboard input unit is therefore now given with specific reference to the perspective view of the calculator as in Figure 1 and the plan view of the keyboard as in Figure 8, except as otherwise indicated.

Line Switch

An on-off line switch 182, which may be considered as part of the keyboard input unit, controls the application of power to the calculator and hence initiation of the control signal POP from the power supply.

As shown in Figure 2, the calculator may be operated at 240, 220, 120, or 100 volts +5%, -10% as determined by a pair of line voltage selector switches mounted at rear panel 34 of the calculator housing and at a line frequency within the range of 48 to 66 Hertz. The calculator is provided with a 6-amp fuse and either a 1-amp fuse for operation at a line voltage of 220 or 240 volts +5%, -10% or a 2-amp fuse for operation at a line voltage of 100 or 120 volts +5%, -10%. It is also provided with a three-conductor power cable 184 which, when plugged into an appropriate AC power outlet, grounds the calculator housing. The maximum power consumption of the calculator is 150 voltamps. No more than a total of 610 voltamps may be drawn from AC power outlets 42 provided for peripheral units.

EXECUTE

The EXECUTE (often referred to as EXEC) key, when pressed, will perform the indicated operations previously keyed in, if any, and display the result of any arithmetic statements on the 32-character display. (Although the display is only 32 characters, an 80-character line can be keyed in with automatic scrolling both for programs and for keyboard operations.) Most keys, when pressed, immediately cause their mnemonic to be displayed. However, pressing certain keys, such as PRT ALL (to be discussed later), allows a particular mode to be in effect till that mode is overridden.

FIXED N, FLOAT N

Immediately after either turn-on or SCRATCH EXEC, the user read/write memory area is cleared; numerical calculations that are executed will be displayed in float-nine notation. The values 2 and 12 in float-nine notation would appear as 2.000000000E+00 and 1.200000000E+01, respectively. Float 9 refers to the nine digits succeeding the decimal point; E symbolizes $\times 10$ raised to the power of the two digits following the E.

NOTE

In this text, individual keyboard operations will be identified by being underlined; e.g., 3 $\pm$ 2 EXEC. (On the display would appear 5.000000000E+00.)

You can specify the desired notation by pressing FIXED N or FLOAT N followed by the appropriate number from 0 through 11. The designated 'N' indicates the number of digits to be displayed to the right of the decimal point after execution.

For example:

FIXED N 7 EXEC,
then 123456.7 EXEC, displays 123456.7000000
or, 123456789.123456789 EXEC, displays 123456789.1230000+

FLOAT N 5 EXEC,
then 123.4 EXEC displays 1.23400E+02
or, 1234567 EXEC displays 1.23457E-01++

In fixed-n notation, a maximum of 12 digits will be displayed to the left of the decimal point; beyond that value the calculator reverts to float-n notation, with the number of digits displayed to the right of the decimal point determined by the particular fixed-n.

The FIXED N and FLOAT N keys are not programmable; that is, they can be used only in keyboard operations. Output formatting under program control is accomplished by the format statement, which is discussed below.

The calculating range of the calculator is: $-9.9999999999 \times 10^{99}$ through -10^{-99} , 0, and 10^{-99} through $9.9999999999 \times 10^{99}$.

CLEAR, DELETE LINE

Pressing either CLEAR or DELETE LINE will erase whatever previously had been displayed during keyboard operations. Pressing either key will cause — (the Lazy T) to appear on the far left of the display, indicating that the calculator is available for new inputs. The difference between the two keys occurs when a stored program line is displayed; DELETE LINE will erase the line from the stored program, whereas CLEAR will only erase the display and not affect the program line itself. (A program line can also be deleted by keying in the line number followed by END OF LINE.)

RECALL

Pressing RECALL during keyboard operations allows the last line that was

*The machine calculates to 12 significant digits regardless of the display length.
**Since only five digits can be displayed to the right of the decimal point the fifth digit is rounded.

executed to be recalled to the display; pressing RECALL during program-inputting operations allows the last program line that was input to be recalled to the display. If an error message appears on the display when an attempt is made either to execute a line in keyboard mode or to input a program line into memory, pressing RECALL will allow the original line to be reviewed for editing purposes.

RESULT

Pressing CLEAR RESULT EXEC displays the numerical value of the last arithmetic statement that was executed. The RESULT key not only allows the result of the previously executed statement to be reviewed, but also can function as an "accumulator" during arithmetic operations; e.g., by pressing

$2 \div 4$ EXEC (in FIXED 2 notation) displays 6.00, then pressing
 $3 \div$ RESULT EXEC displays 9.00, then pressing
 $4 \div$ RESULT $\div$ RESULT EXEC displays 22.00.

Keying in either R E S or R E S U L T has the same effect as pressing the RESULT key.

PRINT ALL

By pressing PRINT ALL, you can determine whether or not the calculator is in print all mode. If ON appears, both the expression and the result of any executed calculation will be typed on whichever printing device is plugged into the calculator; if OFF appears, the information will appear only on the display. Pressing the PRINT ALL key a second time causes the alternative mode to be in effect. In the print all mode, each program statement is printed when END OF LINE is pressed. All error messages are also printed.

BACK, FORWARD, INSERT

The BACK and FORWARD keys can be used to edit expressions on the display. Successive presses of the BACK key will move a blinking cursor to the desired location within the display. Editing can then be performed at this location. The FORWARD key performs the same function as the back key, but in the opposite direction.

At the location of the blinking cursor, the following editing can be performed:

1. A character can be inserted by pressing INSERT. (This opens up a space to the left of the cursor, thereupon moving the cursor to the location of the space; additional presses of the INSERT key will open up more spaces, with the cursor always positioning itself in the left-most space, thus allowing for the immediate insertion of more than one character.)
2. A character can be changed by overscoring it with another character.
3. A character can be deleted by pressing either the space bar or SHIFT INSERT, the only difference being that pressing SHIFT INSERT will close up the space of the deleted character.

Holding either BACK or FORWARD down for approximately 1.5 seconds will cause the cursor to move in rapid succession in the chosen direction.

Once a line is appropriately edited, it can be immediately executed without having to move the cursor to the end of the line.

When a line that is greater than 32 characters (80 characters maximum) is being input, the characters to the left of the display are pushed out of the display region to make room for the additional characters. To view the beginning of the input, press $\rightarrow$ (the right arrow); this operation moves the characters in the display to the right. Pressing $\leftarrow$ (the left arrow) performs the reverse operation. Either arrow key, when held down for approximately 1.5 seconds, will repeat its operation in rapid succession.

ARITHMETIC

There are five basic numerical operators: add (+), subtract (-), multiply (*), divide (/), and exponentiate ($\uparrow$). The order of execution, known as the hierarchy, is identical to the BASIC hierarchy described below. The BASIC functions described below are available on the Calculator simply by keying in the appropriate mnemonics. In addition, the value of π can be obtained by keying in PI.

When operating on trigonometric functions, the calculator assumes the angle to be in radians unless otherwise stated. To express an angle in either degrees or grads, first clear the display, then press either DEG EXEC or GRAD EXEC, respectively; then key in the expression. To revert to radians, clear the display, then press RAD EXEC. Radians, degrees, and grads are also programmable commands.

VARIABLES

The simple "scalar" variables are A through Z and A0 through Z9 (286 total). Simple variables can be used in keyboard operations; e.g., pressing A 3 = 7 EXEC assigns the value of 7 to A3. Unless the value of A3 is then changed or erased from memory, pressing 4 * A 3 EXEC will display 28. If a variable is undefined, any attempt to use this variable in an expression (other than by assigning a value to it) will result in an error message.

An array is an ordered collection of numerical data. An array (subscripted) variable can have either one or two dimensions as indicated by the subscripts, which are presented as numbers within parentheses. A(m) is a one-dimensional array (or column vector) when m designates the row of the element; A(m, n) is a two-dimensional array where m designates the row and n designates the column of the element.

The maximum size of an array is limited by the available calculator memory. At normal 12-point precision, the effective memory limitations would be approximately a (30, 30) array.

Arrays should be referenced in either a common statement or a dimension statement before use in a program; if not, the program defaults to either a 10 element array for a single subscripted array, or a 10 by 10 array for doubly subscripted arrays.

TAPE CASSETTES

The present Calculator is capable of operating with 10 tape cassettes. One cassette is available through the built-in (internal) cassette drive. Four peripheral cassette drives can be connected directly to the calculator through the four I/O slots in the rear panel.

Five more peripheral cassette drives can be added if the user has an I/O expander box (in all, 11 peripheral devices can be added by having an expander box).

SYNTAX

The tape cassette commands have general syntactical rules which will be briefly described. The general command form, with minor variations, is:

COMMAND [UNIT] [FILE], or
COMMAND [UNIT] [FILE] [LNX₁] [LNX₂]

Brackets [] indicate that the enclosed information is optional; parentheses () indicate that the enclosed information is required with the particular command.

COMMAND — The various commands will be individually discussed.
UNIT — Individual units are referenced by the number sign, #, followed by the select code; the internal cassette is designated by #10 (if no select code is given, the internal cassette is assumed); the nine peripheral cassettes are identified by #1 through #9.
FILE — Files within individual cassettes are identified by file numbers; if no file is identified by number, the scratch-pad (or default) file, file 0, is assumed.
LNX₁ — indicates the beginning line number to be affected by the command.
LNX₂ — indicates either the line number where program execution is to begin following the command implementation, or the ending line number to be accessed in the command. In any command whose syntax allows LNX₁ and LNX₂, in order for LNX₂ to be designated, LNX₁ must have been specified.

All information to be input following the command must be separated by commas.

In keyboard mode, press EXEC after fulfilling the syntax requirements of a particular command to implement the command.

PREPARING A FRESH CASSETTE

To prepare a fresh cassette (one which has no markings), first open the cassette door by pressing downward on the switch on the far right of the keyboard; then insert the cassette (print-side up) into the slot built into the door; be certain that the tape is wound around the left spindle — if it is not, but you wish to prepare that side of the cassette anyway, simply close the door and press REWIND.

To store information onto the cassette, first mark the files that are to be used.

PROGRAMMABLE COMMANDS

The following tape cassette commands are programmable: mark, store, load, merge, store key, load key, store data, load data, load binary, rewind, find, and tlist. These commands can also be used in the keyboard mode. The secure command, on the other hand, can be used only in the keyboard mode.

MARK

The mark command produces the designated number of files and defines the file lengths (by the number of 16-bit words per file).

Syntax: MARK [UNIT] (No. of FILES) (LENGTH)

e.g., MARK #10, 5, 1000 EXEC — since unit #10 identifies the internal cassette, including it is superfluous; the '5' indicates the number of files to be made available; the '1000' signifies the number of 16-bit words per file.

Successive files on the tape can be marked with different word lengths as shown in the following example; MARK 3, 1000 EXEC then, MARK 2, 2000 EXEC will mark files 0, 1, 2 of the internal cassette with 1000-word lengths and will mark files 3, 4 of the same cassette with 2000-word lengths.

The length of a file can be changed; however, changing it will affect all the files following it by distorting their contents. To change the length of a particular file, first position the cassette at that file by using FIND (FIND will be discussed more thoroughly, later; then mark the file with the desired length. For Example: to mark file 12 of the internal cassette with 1000 words, press

FIND 12 EXEC then press,
MARK 1, 1000 EXEC

To mark files not previously marked (virgin files), first perform a TLIST; TLIST will be discussed thoroughly later; but for now, merely knowing that TLIST [UNIT] EXEC will list all the marked files, is sufficient. To mark virgin files, it is first necessary to mark the last file listed in TLIST*. As previously discussed, this file is located by the find command. Beginning with this file, successive virgin files can be marked by the methods previously discussed.

To mark the beginning of a tape, be certain that the tape is completely rewound.

STORE

The store command will take the program line numbers in read/write memory, and put them in the designated tape location where they will be saved.

Syntax: STORE [UNIT] [FILE] or
STORE [UNIT] (FILE) [LNX₁] [LNX₂]]

e.g., STORE #2, 3 EXEC; select code # designates which cassette is being accessed; the program will then be stored on that cassette in file number 3.

In the store command, LNX₁ and LNX₂ are used to store a portion of the program: LNX₁ specifies the beginning line number to be stored; LNX₂ specifies the ending line number to be stored. If LNX₂ is not specified, the last line to be stored is assumed to be the highest-numbered program line.

For LNX₂ to be given, it is always necessary to have LNX₁; this is true for any cassette command. In addition, whenever LNX₁ is to be given, the file to be accessed must be identified; e.g.,

STORE 3, 60, 150 EXEC; file 3 of the internal cassette is located; program lines 60 through 150 in memory will then be stored into file 3.

*The number of files referenced in TLIST is always one more than the number marked by the user.

Note

Once a cassette has been marked, any file that has been marked can be accessed by giving its designated file number in a command; e.g., if five files are marked, information can be stored in file 4 even though file 3 is a virgin (empty) file, by pressing STORE 4 EXEC.

LOAD, LINK

The load command will take a program that is stored on a cassette and put it into the memory area.

Syntax: LOAD [UNIT] [FILE] or,
LOAD [UNIT] (FILE) [LNX₁] [LNX₂]]

e.g.,

LOAD EXEC assumes the internal cassette, default file (file 0) is to be loaded into the calculator; any program previously in the memory will be erased.

LOAD 5, 40, 10 EXEC locates file 5 on the internal cassette; the entire program on this file is renumbered beginning at line number 40, and then the program is loaded into memory; program execution is initiated at line number 10. All program line numbers beginning at line 40, that were previously in memory, will be erased and replaced by the program being loaded; if the memory previously had line numbers 10, 20, 30, it will retain them.

In both the load and the merge command, the use of LNX₁ and LNX₂ can have various results. Rules for predicting the results are given at the conclusion of the merge command discussion.

By substituting LINK for LOAD in the previous syntax, the user can implement the link command. This command operates identically to the load command with one exception:

During program execution, if a load statement is encountered, the calculator functions as though RUN EXEC were pressed — that is, the old symbol table is destroyed and a new symbol table is built; on the other hand, if a link command is encountered, the calculator functions as though CONT EXEC were pressed — that is, all variables retain their previous values.

MERGE

The merge command attempts to take program line numbers from the cassette and position them in read/write memory in front of the program currently there, between consecutive line numbers in the program currently there, or behind the program currently there. However, if any line number of the program to be entered matches a line number currently in the program, an error will result. In addition, if the line numbers of the two programs are interwoven, an error will occur; e.g., if the program currently in memory has line numbers 10, 20, 30, 40 and if the program to be merged has line numbers 15, 25, 35, the merge command will cause an error message to occur even though no two line numbers matched.

Syntax: MERGE [UNIT] [FILE] or,
MERGE [UNIT] (FILE) [LNX₁] [LNX₂]]

e.g.,

MERGE #2, 1, 200, 100 EXEC — file 1 of the cassette with select code #2 is located; the entire program on this file is renumbered beginning at line number 200 (LNX₁) and then it is combined with the program currently in memory. Following implementation of the command, program execution will begin at line number 100 (LNX₂).

Either the merge or the load command can be used for stacking programs in read/write memory. The major difference the two commands is as follows: LOAD will erase the line numbers previously in memory, beginning at the designated LNX₁; MERGE, on the other hand, will retain all line numbers previously in 9830A memory.

In both the load and the merge command, LNX₁ and LNX₂ are predictable. Regardless of the mode, LNX₁ renumbers the program line numbers of the accessed file to begin at LNX₁; the spacing between consecutive line numbers remains the same; all GO TO statements, etc. are properly adjusted to reflect the new line numbers; this program is then loaded into user memory.

In program mode:

1. If LNX₂ is given, program execution will continue at LNX₂.
2. If LNX₂ is not given, program execution will continue either at the next higher line number of the original program or at LNX₁, whichever comes first.

5

In keyboard mode:

1. If LNX₂ is given, program execution will begin at LNX₂.
2. If LNX₂ is not given, the calculator will halt after loading in the program.

5

STORE KEY, LOAD KEY

10

The store key command will take all user definable keys (upper left-hand region of the keyboard) that have been defined and put them on a cassette file, which will be tagged as a key file.

10

The load key command takes the user definable information from the cassette and positions it in memory such that each user definable key performs the same operation that it previously did before being stored on tape.

15

Syntax: STORE KEY [UNIT] (FILE)
LOAD KEY [UNIT] (FILE)

15

Spacing may arbitrarily be left between the words STORE and KEY and the words LOAD and KEY. In general, the HP Basic language ignores blank spaces (except, of course, in a quote field where every character and space is duplicated).

20

STORE DATA, LOAD DATA

20

The store data command takes a block of data from memory and puts it on a cassette. Normally, only arrays can be stored using this command; however, if no array is specified in the command, all data in the common statement of the program can be stored. The calculator allows simple variables in the common statement, as well as arrays; in fact, the common statement can accept simple variables, array variables, integer arrays and variables, and split arrays and variables*.

25

25

The load data command takes the data that was previously stored on a cassette file, and loads it into mainline memory. If an array has been stored, then LOADDATA must specify an array; if LOADDATA does not specify an array, an error will result. If, on the other hand, the common area will be retrieved by the load data command (in this case, no particular array can be specified in the command, lest an error occur).

30

30

Syntax: STOREDATA [UNIT] (FILE) [ARRAY]
LOAD DATA [UNIT] (FILE) [ARRAY]

35

e.g.,

35

STORE DATA 6, B EXEC will locate file 6 on the internal cassette; then the B array in the current program will be stored in this file. (A simple variable cannot be stored in this manner).

40

LOADDATA 6, B EXEC can then retrieve the B array and load it into memory whenever it is needed; the following command could be given, also:

40

LOADDATA 6, C EXEC; this would retrieve the B array and load it into memory in place of the C array, provided B and C are the same size and type.

Assume the common statement in a program looks like this: 1 COM A(8), B(5,5), D3, E.

45

45

Pressing: STORE DATA 2 EXEC will store all the common statement variables into file 2 of the internal cassette. Both the array variables and the simple variables will be retrieved by pressing: LOAD DATA 2 EXEC.

Pressing: LOAD DATA 2, A EXEC is illegal and causes an error since no particular array can be retrieved from a file if it was stored in common.

50

50

In all store and load data commands, the file to be accessed must be identified, even if it is the default file (file 0).

*The common statement acts like a dimension statement with the additional feature that data in common is saved from program to program.

45

LOAD BIN

The load binary command will transfer binary information — assembly language program — from the cassette to the user memory. The assembly language program may be a system diagnostic, an I/O subroutine, or a simulated “option block” designed to perform some specific function.

The assembly language program cannot be listed or displayed.

Syntax: LOAD BIN [UNIT] (FILE)

Note

Files on cassettes are tagged as: program files, key files, data files, or binary files. If an attempt is made to load from a particular file and the load command incorrectly identifies the file tag, an error will occur; e.g., pressing LOAD KEY 1 EXEC when file 1 is a program file, causes an error message to appear.

REWIND

Pressing the REWIND key, located on the right-hand side of the keyboard, immediately rewinds the internal tape cassette to the clear leader.

To rewind any other cassette, REWIND must be typed in, followed by the select code of the particular cassette. (The internal cassette can also be rewound by typing in REWIND EXEC.)

REWIND must be typed if the command is to be used in the programming mode.

Syntax: REWIND [UNIT]

e.g.,

REWIND #2 EXEC will rewind the cassette with select code #2.

FIND

The find command (previously mentioned in conjunction with the mark command) is used to locate a particular file. While the cassette is searching for the file number, a “lazy T” appears on the display. During this interval, the cassette is searching under interrupt control, thus returning control of the calculator keyboard to the user. This feature allows for the execution of one portion of a large program, while another portion is being found — thereby improving access time.

When the specified file is found, the cassette tape halts.

Syntax: FIND [UNIT] (FILE)

e.g.,

FIND #3, 2 EXEC causes the cassette with select code #3 to search until file number 2 is located.

TLIST

Beginning with the current tape location, this command reads all subsequent file identifiers and prints out information concerning each file.

Syntax: TLIST [UNIT]

The information for each file, on the designated cassette, is printed out on one line. There are no column headers identifying the information in each line; the assumed headers are:

File No.	File Type* Code No.)	Absolute File Size (in words)	Actual File Size (in words)	Program Line Nos. (Beginning) (Ending) (LNX ₁) (LNX ₂)	Common Area (in words)
----------	-------------------------	-------------------------------------	-----------------------------------	--	---------------------------

*The code numbers identifying the file types are as follows:

- 1 binary
- 2 data
- 3 program (source)
- 4 key

In addition, if the file is secured, the number 2 appears in front of the code number (this applies only to binary, source, and key files); e.g., if "24" appears in the second column, the file is a secured key file.

If a file is a data file, LNX₁ is superfluous; in this case the data is described in this column as:

- 0 full precision
- 1 split precision
- 2 integer precision
- 3 common

If the file is not a program file, the last two columns will contain no information.

NON-PROGRAMMABLE COMMANDS

SECURE

This command has the capability of concealing program lines from potential users; that is, your program could be given to another, and that person could load and run it — however, he would not be able to fetch particular program lines for viewing purposes nor could he store the program on any other cassette file.

An attempt to fetch a secured program line will result in the line number appearing on the display, followed by an *; attempts to list the program will result in the secured line numbers appearing followed by an *.

If any lines in a program are secured, the entire program is considered to be secured; that is, even though certain program statements are visible, none of the program can be reproduced onto another cassette file.

Syntax: SEC [LNX₁ [LNX₂]] or SECURE [LNX₁ [LNX₂]]

It is therefore, possible to secure specific lines within a program; e.g., pressing:

SEC 30, 80 EXEC, followed by

STORE 2 EXEC secures lines 30 through 80 of the program in memory and then stores both the secured and unsecured portions of the program into file 2 of the internal cassette.

When a program is initially secured, it can still be reproduced onto as many files as necessary; however, once the program is scratched from memory, (even though it can be loaded back into memory) it cannot be reproduced onto any cassette files.

When program lines are secured, the entire calculator is in the secured mode. Therefore, after the secured program is stored away, the user should press SCRATCH A before inputting other programs — thus, avoiding "secured program" errors.

User definable keys (when not being used as typing aids) can be secured too. Just press FETCH (particular key) SEC EXEC and the designated key will be secured.

Note

To protect all the information on a particular cassette, break one of the tabs on the top of the cassette; this makes the cassette inaccessible for further storage.

PROGRAMMING

The programming language of the calculator is, with minor variations, BASIC as described below.

Instructions to the computer within a program are provided by program statements. Each statement in Basic has an associated line number which must appear in the left-most portion of the statement.* Statement line numbers appear in ascending order with 9999 being the largest possible line number.

A program statement, which has been correctly keyed in, can be stored in read/write memory by pressing the END OF LINE (EOL) key. This key is situated in an area corresponding to the carriage return/line feed key on a teletype (R.T.M.) keyboard.

*The length of a statement, including the line number and including appropriate spacing, can be up to 80 characters.

AUTO

As previously mentioned, statement line numbers can be automatically input. In its simplest form, pressing AUTO # EXEC causes line number 10 to immediately appear on the display awaiting the program statement. The line numbers of additional statements will be in ascending order with a spacing of ten between consecutive line numbers. The AUTO # (AUTO) syntax and the examples to follow present some of the alternatives in automatic line numbering.

Syntax: AUTO # [LNx, [Spacing]]

LN_x is the beginning line number to be automatically input; the desired spacing between lines can then be input if LN_x is given. If no spacing is indicated, a spacing of 10 is assumed.

e.g., AUTO # 30 EXEC AUTO # 40, 2 EXEC AUTO # EXEC

30	40	10
40	42	20
50	44	30
:	:	:

When AUTO # is pressed, AUTO appears in the display.

Note

Although the CLEAR and DELETE LINE keys have previously been discussed, the following point should be made. If a program line currently being keyed in is found to be totally unacceptable (not worth salvaging by using the editing keys), it can be erased by using either the CLEAR or the DELETE LINE keys. However, if the program line numbers have been automatically input, pressing CLEAR not only erases the entire display but also eliminates the AUTO # mode, whereas pressing DELETE LINE erases only the program statement without affecting the line number itself.

PROGRAM VIEWING

There are two methods of viewing a program that is currently in memory:

1. List it on a printing device.
2. Bring it line-by-line to the display.

LIST

The list command has two specific applications: it can be used to provide a total listing of the programs in read/write memory, or it can be used to indicate the read/write memory available for inputting; e.g.,

LIST EXEC lists all program lines that are in memory on the user's standard printing device.

LIST #3 EXEC lists all program lines that are in memory on the peripheral with select code #3.

LIST 9999 EXEC causes the number of 16-bit words available in memory to be displayed.

↓,↑

If a specific program line is in the display, pressing ↓ (down arrow) displays the next higher-numbered program line;↑ pressing ↑ (up arrow), on the other hand, displays the next lower-numbered program line. When no program line is currently in the display, pressing ↓ would display the successively higher line from the one most recently displayed; whereas ↑ would display the successively lower line.

*It should be noted that if a program line has more than 32 characters, the first 32 characters will be displayed; by pressing ←, the rest of the line can be viewed as the display is scrolled to the left. At the end of the line — appears.

FETCH

In addition to viewing a program line-by-line, specific program lines can be immediately brought to the display by using the fetch command.

Syntax: **FETCH [LNX₁]**

5 Where LNX₁ is the specific line number to be accessed;

e.g.,

FETCH EXEC always displays the lowest-numbered program line.
FETCH 300 EXEC displays line 300 if it exists; if line 300 is not available (and there are other higher-numbered lines), the line immediately higher than 300 will be displayed; if there are no line numbers as high as 300, the highest-numbered line available in memory will be displayed.

PROGRAM EDITING

Any displayed program line can be edited by using the **BACK**, **FORWARD**, and **INSERT** keys.

15 The **RECALL** key, previously discussed, need only be briefly mentioned. A program line is keyed in; if an error message appears on the display when **EOL** is pressed, the program line can be reviewed by pressing **RECALL**. Appropriate editing can then be performed on the program line. The previously input line can always be recalled by pressing **RECALL** whether or not an error appears.

20 **CLEAR** and **DELETE LINE** have already been thoroughly discussed. However, to reiterate, there are two methods of deleting a program line currently in memory:

1. If the line is currently in the display, pressing **DELETE LINE** will erase it from memory (**CLEAR** only clears the display).
- 25 2. Any line in memory can be immediately deleted by keying in the appropriate line number followed by **EOL**.

DELETE

The delete command (to be distinguished from **DELETE LINE**) can selectively delete program lines.

30 Syntax: **DELETE [LNX₁ [LNX₂]]** or **DEL [LNX₁ [LNX₂]]**

Where LNX₁ is the first line to be deleted and LNX₂ is the last line to be deleted:

e.g.,

DELETE EXEC deletes all program lines;
DELETE 40 EXEC deletes all statements beginning at line number 40;
DELETE 50, 80 EXEC deletes all lines numbered 50 through 80.

SCRATCH

The scratch command can erase a variety of things from memory:

e.g.,

40 **SCRATCH EXEC** erases all program lines and variables;
SCRATCH A EXEC erases everything from memory — program lines, user definable keys, variables (identical to turning the calculator off, then on again);
SCRATCH K EXEC erases all user definable keys (user definable keys will be discussed later);

45 **SCRATCH V EXEC** erases all variables;
SCRATCH (particular UD key) erases the particular user definable key that was pressed — pressing **EXEC** is not required in this case.

Scratch can be accessed either by keying in the seven letters or by pressing **SCRATCH**.

RENUMBER

50 The renumber command will take all the program line numbers and renumber them.

Syntax: **RENUMBER [LNX₁ [SPACING]]** or **REN [LNX₁ [SPACING]]**

Where LNX₁ will be the new line number of the first program statement, and SPACING will be the spacing between consecutive line numbers:

e.g.,

REN EXEC will renumber all statements by numbering the first statement '10' with a spacing of 10 between statements;

RENUMBER 30 EXEC rennumbers the first statement '30' with a spacing of 10 between statements;

REN 45, 20 EXEC rennumbers the first statement '45' with a spacing of 20 between statements.

All statements in the program that reference another line number are appropriately corrected with the renumber command; e.g., GO TO 80 would be corrected to reference the line that replaced 80.

PROGRAM DEBUGGING

NORMAL, TRACE

The TRACE key can be used to determine the order of statement execution for a program that is currently running. Pressing TRACE during program execution causes the line numbers to be printed in the order in which they are accessed; then pressing NORMAL reverts the calculator to the normal mode. Thus, when a program is running, both TRACE and NORMAL are immediate execute keys.

When no program is running, to revert to either trace or normal mode requires pressing the appropriate key followed by EXEC. Trace mode can be set up to trace specific line numbers.

Syntax: TRACE [LNX₁ [LNX₂]]

Where LNX₁ is the first line number to be traced, and LNX₂ is the last line number to be traced;

e.g.,

TRACE 20 EXEC will trace beginning at line number 20 when the program is running.

TRACE 50, 60 EXEC will trace beginning at line number 50 and ending at line number 60, each time these line numbers are executed in the program.

STOP

The stop command can be a statement within a program (to be discussed later) and can be used as a debugging tool.

As a debugging tool, STOP is extremely valuable. A program that is running can be stopped at any time by pressing STOP (the current line number of the program will be displayed). If any program lines are then edited, the program must be rerun from the beginning, using the RUN key. If no editing has been performed, the program can continue where it left off if the CONT key (to be discussed in detail later) is pressed.

While a program is stopped, the values of variables can be checked to determine if the program is doing what was intended; e.g., pressing A EXEC would determine the present value of the simple variable A.

While a program is stopped, STOP⁺ can have another function. Pressing STOP displays STOP; then keying in either one line number or two line numbers separated by a comma — indicates that the calculator should stop program execution at these line numbers. Pressing CONT EXEC will then start program execution; e.g., STOP 80 EXEC then CONT EXEC or RUN EXEC will cause the program to halt at line number 80. Once a program is running under these conditions, there is one way to revert to normal program execution: After the program has halted, press STOP EXEC, then CONT EXEC, and the program will no longer stop at the given line numbers.

A line or lines can also be designated by the user in similar manner and prior to the execution of a program and without altering the program; before execution of which program execution is halted.

⁺When a key is not being used as an immediate execute key, whether the key itself is pressed or whether the letters displayed on the key are individually keyed in, is generally arbitrary.

STEP

After a program is halted by a stop command, execution can continue by pressing STEP. STEP is always immediate execute; it causes the program to execute the appropriate statement and then to halt. Therefore, after each statement is executed, it can be checked to ensure that it performed the required function. When a particular function is considered satisfactory, either CONT or STEP can be pressed; CONT will execute the rest of the program while STEP will execute the next program statement only.

RUN, CONTINUE

Pressing RUN EXEC causes a program to begin execution at the first statement regardless of whether the program had previously been halted by a stop command; however, if the program had been halted by a stop command, pressing CONT EXEC will begin program execution where it had previously halted.

As previously mentioned, if a program is edited after it has been halted, program execution is reinitialized at the first line by RUN EXEC*. However, if, during the halt, the values of variables are changed or other internally-programmed conditions are changed, then CONT EXEC must be pressed to keep these newly adjusted conditions intact. The following things can be done to the program while it is halted if, upon completion, the program is executed by pressing CONT EXEC:

1. Variables can be changed; e.g., B = 5 EXEC sets the simple variable B equal to 5.
2. Angular units in trigonometric functions can be changed to measure in radians, degrees, or grads, depending on the user's requirements: e.g., DEG EXEC will assume all angles to be in degrees.
3. Write, print, and display statements can be input from the keyboard (these statements will be discussed later).
4. The data pointer, which indicates the next datum to be encountered, can be reset to the beginning of the data by typing RESTORE EXEC.
5. The program can go to a particular statement and be available for execution there; e.g., GO TO 80 EXEC sets the program line counter to 80 for either step-by-step or continuous execution. (If continuous execution is desired at line number 80, the continue command can be employed, as discussed below.) IF... THEN can also access a particular line number.
6. Any calculator-keyboard statements can be executed.

A halted program can be executed beginning at any line number by using the continue command; e.g., CONT 95 EXEC will continue execution starting at line number 95.

A program can be run beginning at any line number; e.g., RUN 110 EXEC will begin program execution at line number 110.

NOTE

The major difference between RUN and CONT is that RUN initializes all variables in the program and reverts to all normal program modes, while CONT neither affects any variables nor affects any current program modes.

PROGRAM STATEMENTS**LET**

As in Basic LET A = 6 is a legitimate statement; however, the implied let statement is also allowed. Thus, A = 6 is the same as LET A = 6.

GO TO, GO SUB

The GO TO and GO SUB statements are the same as in Basic, however, each statement has one additional feature called respectively, the computed GO TO and the computed GO SUB. In either case an expression is evaluated and the rounded integer value of the expression is determined; the integer then acts as a pointer to a particular line number from a parameter listing in the statement. Some examples should explain this feature more clearly:

(In all examples, the present value of T will be 2.)

*The initialize command, to be discussed later, is also legitimate.

SI

20 GO TO T12—3 OF 250, 350, 450

Since the integer value of the expression is one, the first parameter following "OF" will be accessed (line number 250).

80 GO SUB T+2.5 OF 130, 260, 330, 370, 490

5 The rounded integer value of the expression is 5; therefore, the subroutine beginning at line number 490, the fifth parameter following "OF", will be accessed. (Decimal values of .5 and above are always rounded) to the next higher integer value.)

10 Any legitimate expression can be used; if the rounded value is either less than 1 or greater than the number of parameters following "OF", then the line number following the GO TO or GO SUB statement is executed.

PRINT

15 The calculator has one feature in the print statement not generally available in BASIC. Alphabetic information in a quote field can be printed in either upper or lower case letters. Printing in lower case is just the opposite of that on a regular typewriter; with shift or shift lock pressed, letters inside the quote field will be printed in lower case (the display, however, will still appear in upper case). If the "at" symbol (ⓐ) is required, press SHIFT RESULT.

20 Quote fields in both the format and write statements also have this "lower case" feature available.

DISPLAY

25 The display statement performs the same function as a print statement; the difference is that the information appears on the display rather than on a printing device. Thus, when a permanent record of the information is desired, the print statement should be used. The syntax used is DISP.

FORMAT, WRITE

30 The format statement is a means of structuring program printers in a specified manner. The write statement defines the variables, constants, etc. that will appear on the printout; it also determines the device to be printed upon and the particular format statement to be followed. The following examples should adequately explain the features in both the format and the write statements:

e.g.,

5 FORMAT F10.2

6 WRITE (15, 5) .7

35 printout will be ^^^^^^0.70 where the "^" indicates a blank space.

40 F10.2 — the "F" refers to fixed-point format; "10" refers to the total field width reserved for the printout "2" refers to the number of digits to the right of the decimal point. Excess space to the right of the decimal point will be filled with zeros; one space is reserved for the decimal point; two spaces are reserved to the left of the decimal point, one for a digit preceding the decimal point, another for a sign (however, only minus signs are printed).

45 (15, 5) — in the write statement, the information within parentheses is required. "15" refers to the printing device to be used (select code 15 refers to the standard printer); "5" refers to the line number of the format statement that is being accessed (in this case line number 5). The format statement, which is being accessed, can appear anywhere in the program listing.

50 .7 — in the write statement, the information following the right parenthesis is to be printed according to the designated formats. The value is always right-justified within its field width.

e.g.,

20 A = 62.4

25 FORMAT E12.2, F7.3, X, E8.1, /, 2F4.0, 3X, E9.0

260 WRITE (15, 25) 231, -61, -12.4, A, "IS THE VALUE OF A", .4, A+3

printout will be:

231-61-12.4 62.4 IS THE VALUE OF A 0 7.E+01

The format statement, the write statement, and the printout all have individual fields referenced; interlinking fields are represented by corresponding reference numbers.

Since the write statement references the format statement numbered 25, values in the write statement will be formatted as specified in this format statement.

REFERENCES

- 1 E12.2 indicates exponential (floating-point) notation with a field width of 12 and two digits to the right of the decimal point. The printout displays the exact form. Remember the field width must be large enough to include a leading sign, the decimal point and E+XY. Since printouts from the write statement are right justified and since "231" takes up only 8 of the 12 character field width, the four blank spaces are to the left of the value.
- 2 The value "-61" in the write statement totally fills the F7.3 field, therefore there is no space between this value and the previous value in the printout. Since F7.3 indicates three digits to the right of the decimal point, zeros are supplied in this case.
- 3 "X" indicates a space between values; therefore, the values supplied for the F7.3 and E8.1 formats will be separated in the printout by at least one space.
- 4 The value, -12.4 totally fills up E8.1, in fact, the last digit is suppressed since, with this field designation, only one digit can follow the decimal point.
- 5 "/" tells the printing device to skip one carriage return to the beginning of the next line.
- 6 2F4.0 specifies two consecutive fixed-point formats of F4.0 to be used for values in the write statement. The first format is for the value of A, which from program line number 20 is 62.4; when a fixed-point format specifies zero digits to the right of the decimal point, the value supplied is rounded to be an integer and the decimal point is suppressed — 62, in this case. Since the carriage return had previously been specified, this value is printed on the beginning of the next output line.
- 7 Quote fields are printed in the sequence in which they occur. Since this quote field is in the write statement, it is printed immediately after 62.4 is printed in F4.0 format. Note the space in front of "IS"; without this space, the printout would read 62IS instead of 62 IS. Quote fields can appear in either the format or write statements.
- 8 The value, .4 is also to be printed in F4.0 format. The rounded integer value of .4 is 0 — hence, the printout.

9 "3X" indicates there should be three spaces between the values supplied for the F4.0 and the E9.0 formats.

10 Expressions can be specified in write statements. The value of A+3 is 65.4; however, with this format, it will be rounded to 7.E+01. The decimal point is not suppressed in floating-point notation when zero is specified as the number of digits to the right of the decimal point.

If there are more values presented in the write statement than there are formats in the referenced format statement, the formats will be repeated; e.g.,

10 FORMAT F6.2, E10.2

20 WRITE (15, 10) 18, 21, 19.3, 29.6, .71

printout will be:

^18.00^^2.10E+01

^19.30^^2.96E+01

^^0.71

15 After the first two values are printed according to the specified formats, the output printer's carriage return is activated, then two more values are printed according to the same two specified formats, etc.

Formatting Rules:

20 In fixed-point format, Fm.n, m stipulates total field width and n stipulates the number of digits to the right of the decimal point. If $n > 0$, the minimum field width allowable is $m = n + 3$; e.g., F4.1 for a value of -0.6 would print -0.6, which takes up the total field width of 4. If $n = 0$, the minimum field width is $m = 2$; e.g., F2.0 for a value of -7 would print -7, thus taking up the allotted field width.

25 In floating-point format, Em.n, m and n are the same as in fixed-point format. However, the minimum field width allowable is always $m = n + 7$.

The following are all allowed in format statements:

Fm.n — fixed-point formats;

Em,n — floating-point formats (often called exponential or scientific notation);

X — space;

/ — carriage return (for printing device);

"****" — quote field;

B — binary format (where write statement could have octal number, the binary equivalent would be output).

All of the above can be duplicated any number of times by leading the symbol with the appropriate number.

40 The following are all allowed in write statements: constants, variables, expressions, and quote fields. It should be noted that a write statement can be input from keyboard mode; that is, the write statement can reference the line number of a format statement in memory without being in the program itself.

45 The maximum width of both the fixed-point and the floating-point fields is 9,999. However, the programmer is effectively restricted by the allowable characters per line of the printing device.

The information in the format statement must be separated by commas.

50 The information in the write statement must be separated either by commas or semicolons; generally it makes no difference. However, one additional feature of the write statement is that it can perform the identical operations as the print statement; the benefit is the ability of the write statement to select the device to be printed upon. To write on a punched tape photoreader with select code 2, the write statement could be set up in the following manner:

54

30 WRITE (2, *) A; B; C, D

The "*" indicates that no format statement is referenced; thus, WRITE acts like a PRINT statement: data will be left justified, semicolons pack the output fields, commas spread out the fields, etc.

5 P TAPE

As in BASIC, PTAPE causes the computer to read in a program from the punched tape photoreader. If the photoreader select code is 5, then pressing either PTAPE #5 or PTA #5 will perform this task unless no photoreader is connected to the calculator. If a photoreader is not connected, the calculator will wait till one is hooked up to complete the command. During this time the display will be blank.

During the implementation of this command, lines being loaded into memory have their syntax checked; if a line is in error, it will be rejected — thus, only those lines with correct syntax are loaded into the calculator. To obtain a record of the rejected lines, it is necessary to put the calculator in the print-all mode prior to pressing PTA #5; in print-all mode, all rejected lines are printed.

Note

To punch information onto paper tape, use the list command as discussed earlier.

MULTILINE FUNCTIONS

Multiline functions serve the same purpose as single-line functions with the added capability of being able to describe more sophisticated functions.

In single-line functions the general form of the defining function is:

statement number DEF FN single letter A to Z (simple variable)* = expression

In multiline functions, the general form is the same aside from the equal sign and the expression; for in a multiline function, the expression can be spread out over many statement numbers. Thus, an extremely complicated expression can appear in a more simplified manner; additional flexibility is also gained in that the value of any variable within the expression for a given value of the argument of the function; e.g.,

10 W = .5

20 Y = 2

30 PRINT FNA (3)

40 STOP

50 DEF FNA (X)

60 Z = X¹² + Y¹²

70 Q = Z + 3

80 D = Q/W

90 RETURN D

100 STOP

When this program is run, D will return a value of 32. The return statement returns the result of FNA (X) which, in this case, is equal to D. Line 40, the stop statement is needed to keep the calculator from trying to re-execute lines 50 through 90 after D is printed; without STOP in line 40, an error occurs in line 60 since a second pass beginning at line 50, would be made with X undefined.

*The simple variable is a dummy variable which indicates where the actual argument of the function is used in the defining expression.

Any variable evaluated in the expression can be returned by the return statement, and multiple return statements are allowed; e.g.,

10	X = 3	100	RETURN Z		
20	INPUT Y	110	Q = Z+2		
5	30	WRITE (15,900) FNG (Y)	120	IF Q ≤ 100 THEN 150	5
	40	END	130	PRINT "Q ="	
	50	DEF FNG (Y)	140	RETURN Q	
	60	Y = Y + 1	150	PRINT "Z IS"	
	70	Z = Y+2*X	160	RETURN Z	
10	80	IF Z ≤ 100 THEN 110	900	FORMAT F12.1	10
	90	PRINT "Z ="	1000	END	

In this example, a variable can be returned from three different lines (100, 140, 160) depending on the initial value of Y; Z can be returned from both lines 100 and 160 if the value of Y meets certain criteria.

Caution must be taken as to the placement of the statement that calls the function (in both examples, statement 30), and it is generally advisable to put a stop statement immediately after this statement (as in line 40 of both examples); otherwise, an undesirable loop may develop.

If correctly entered, the function of a function can be evaluated.

STOP, END

STOP was previously discussed, with emphasis on its program debugging capabilities. Now it will be discussed as a program statement, emphasizing the differences between it and the end statement.

When the program encounters a stop statement, it halts and is waiting; if CONT EXEC is then pressed, the program will continue with the statement following the STOP. This is not true if an end statement is encountered; the program will halt, but if CONT EXEC is pressed, the calculator will revert back to the lowest-numbered statement in memory. Therefore, the stop statement should be used between stacked programs that are to be run sequentially. When STOP is used in this manner, the values of simple variables can be passed from program to program.

A program should be terminated by encountering either a STOP or an END. The highest-numbered program statement need not be an end statement.

INPUTTING DATA

Program data can be input in three ways: the input statement, the read and data statements, and the initialize command. The input statement and the read and data statements are thoroughly discussed below.

INITIALIZE

The INITIALIZE key, when pressed, allocates storage space in memory for array variables. After the required data is keyed in, the program can be executed by pressing CONT EXEC; remember — RUN EXEC erases the values of all variables, thereby requiring all variables to be defined in the program itself.

Simple variables can always be input in keyboard mode without using the INITIALIZE key, as long as CONT EXEC is pressed to run the program. Since array variables can be input in keyboard mode by using the initialize command, it is not necessary to define any variables in the program itself. It is still necessary, however, to identify arrays in either a dimension or a common statement.

USER-DEFINABLE KEYS

There are ten User-Definable Keys (UDK) in the upper left-hand block of the

keyboard. There are, however, effectively 20 accessible UDK's since each key can be accessed normally or with the shift key held down.

To enter UDK mode, press FETCH (particular UDK); the display will then read, KEY indicating the mode. To exit from UDK mode, press CLEAR END EXEC; the scratch command can also be used to exit from UDK mode — this command will, of course, erase certain information in the process. UDK mode is automatically exited when certain sequences are followed; these cases will be discussed later.

The user-definable keys can be used effectively in three ways:

1. to represent text (where text can be used as a typing aid);
2. to represent functions (where different values can be passed to the function);
3. To represent programs.

REPRESENTING TEXT

If a key represents text, merely pressing the key will immediately display the text without erasing anything that was previously on the display. Thus, commonly used words and phrases can be put on keys to serve as typing aids. Text can be put on a key in the following manner:

First, access a key by pressing FETCH (particular UDK). Then, press * followed by a character string* and finally EOL. Besides inputting the character string, pressing EOL in this sequence takes the user out of UDK mode. Any time the programmer wishes to use a character string, he must press the key into which the desired character string was input.

If, for example, a key was accessed by FETCH (particular UDK); then * FORMAT F10.2,X,E10.1 EOL was input. If subsequently program line number 60 needed this format, pressing 60 (particular UDK) EOL will put line number 60 into memory with the required format.

A typing-aid key can be used as an immediate execute command if an * is placed both in front of the text and following the text:

e.g., FETCH (UDK) *LOADDATA #4, 6, B* EOL

This command will be immediately executed whenever the UDK is pressed.

To use a key that has text, merely press the key. Pressing FETCH (particular UDK) will display the * with the text; however, text can be edited if the fetch command is used. Pressing FETCH and then * will erase the old character string and then wait for new text to be input.

REPRESENTING FUNCTIONS

A user definable key can be used to represent functions — either single or multiline. In either case, after the key is accessed, the function must be preceded by a line number — input either manually or automatically.

After a key has been accessed, the following function could be input:

10 DEF FNA (X) = 7 * X -3 EOL

Whenever a value is to be passed to X (the argument of the function), first press the appropriate key; the display will read FNA. Then key in the appropriate value of the argument, which can be either a constant or an expression (e.g. 20), and press EXEC; the value of the function will then be displayed (in this case, 137). The same result could have been achieved by using the fetch command; however, FNA would not appear automatically on the display; it would have to be keyed in along with the argument; e.g., FETCH (UDK) FNA 20 EXEC would also display 137.

If a multiline function, DEF FNB (Z), has been input, pressing the appropriate UDK causes FNB to be displayed; as before, passing a value to the argument, Z, and then pressing EXEC will compute and display the value of the function.

Functions in mainline memory and in a UDK can be called, regardless of the current operating mode.

*The maximum length for the character string (including the *) is 80 characters.

If a function in the calculator is defined in more than one place, the first function found with the designated name will be accessed. If the user is in UDK mode, the calculator will search for the function in the following order:

- 5 1. The current UDK program will be checked. 5
- 2. The first line of each key (in the order defined) will be checked.
- 3. Mainline memory will be checked.

If the user is not in UDK mode, the order of the search will be steps 3 and 2, respectively.

REPRESENTING PROGRAMS

10 A UDK can be used to represent an entire program. Programming rules in UDK mode are consistent with those discussed above. There is one restriction, however; if a common statement is used, its size must be less than or equal to the size of the common statement in the mainline program — for there is only one common area allocated to memory. 10

15 To run a program that is represented in a UDK, it is advisable to press RUN (particular UDK) or FETCH (UDK), then INIT (particular UDK). The program can be continued merely by pressing the (particular UDK); but if there are array variables in the program, pressing only the key will cause these variables to be undefined (similar in this respect to the continue command, which neither destroys the old symbol table nor builds a new one). After executing the program, the calculator will exit from UDK mode. 15

20 Programs represented on a UDK generally use only simple variables for the obvious ease of handling. 20

To list program lines on a particular UDK, press:

25 LIST (particular UDK); 25

pressing LIST EXEC in the UDK mode will list the program lines on the key currently being accessed. To selectively list particular lines on the UDK, it is first necessary to FETCH a key; then use the list command as discussed below.

30 To load a program from a cassette file onto one particular key, first FETCH the key, then give the load command followed by EXEC. Program lines on a particular key can be stored onto the cassette in the same manner; text (as a typing aid), however, cannot be stored in this manner. 30

STORE KEY and LOAD KEY can be used for any UDK regardless of the information on the key. The use of these keys is discussed below.

STATEMENTS

This is a BASIC statement:

10 INPUT A,B,C,D,E

COMMENTS

A *statement* contains a maximum of 80 characters.
A *statement* may also be called a line.



STATEMENT NUMBERS

Each BASIC statement begins with a *statement number* (in this example, 20):

20 LET S=(A+B+C+D+E)/5

COMMENTS

5 The number is called a *statement number* or a *line number*.
The statement number is chosen by you, the programmer. It may be any integer from 1 to 9999 inclusive.

Each statement has a unique *statement number*. The computer uses the numbers to keep the statements in order.

10 Statements may be entered in any order; they are usually numbered by fives or tens so that additional statements can be easily inserted. The computer keeps them in numerical order no matter how they are entered. For example, if statements are input in the sequence 30,10,20; the computer arranges them in the order: 10,20,30.

INSTRUCTIONS

15 The statement then gives an *instruction* to the computer (in this example, PRINT):

30 PRINT S

COMMENTS

Instructions are sometimes called *statement types* because they identify a type of statement. For example, the statement above is a "print" statement.

OPERANDS

20 If the instruction requires further details, *operands* (numeric details) are supplied (In this example, 10; on the previous page, "S"):

40 GO TO 10

COMMENTS

The *operands* specify what the instruction acts upon; for example, what is PRINTed, or where to GO.

A PROGRAM

The sequence of BASIC statements given on the previous pages is called a *program*.

```
10 INPUT A,B,C,D,E
20 LET S=(A+B+C+D+E)/5
30 PRINT S
```

The last statement in a program, as shown here, is an **END** statement.

```
40 GO TO 10
50 END
```

COMMENTS

The last (highest numbered) statement in a program must be an **END** statement.

The **END** statement informs the computer that the program is finished.

FREE-FORMAT LANGUAGE

BASIC is a "free format" language — the computer ignores extra blank spaces in a statement. For example, these three statements are equivalent:

```
30 PRINT S
30 PRINT S
30PRINTS
```

COMMENTS

When possible, leave a space between words and numbers in a statement. This makes a program easier for people to read.

TERM: SIMPLE VARIABLE

DEFINED IN BASIC AS:

A letter (from A to Z); or a letter immediately followed by a digit (from 0 to 9).

EXAMPLES:

```
A0 B
M5 C2
Z9 D
```

COMMENTS

Variables are used to represent numeric values. For instance, in the statement:

```
10 LET M5 = 96.7
```

M5 is a variable; 96.7 is the value of the variable M5.

There is one other type of variable in BASIC, the array (subscripted) variables; its use is explained in Section IV.

TERM: NUMBER

DEFINED IN BASIC AS: A decimal number (the sign is optional) between 1E-99 and 9.999999999999E+99

Zero is included in this range.

EXAMPLES:

-10008	5	3.14159	10E+37
126.257	0	10 E37	10E-37
16.01	.06784	-10 E37	1.0E+2

TERM: E NOTATION

DEFINED IN BASIC AS: A means of expressing numbers having more than six decimal digits, in the form of a decimal number raised to some power of 10.

EXAMPLES:

1.000000E+06 is equal to 1,000,000
and is read: "1 times 10 to the sixth power" (1×10^6).
1.020000E+04 is equal to 10,200
1.020000E-04 is equal to .000102

COMMENTS

"E" notation is used to print numbers having more than six significant digits. It may also be used for input of any number.

When entering numbers in "E" notation, leading and trailing zeroes may be omitted from the number; the + sign and leading zeroes may be omitted from the exponent.

The precision of numbers is 12 decimal digits.

TERM: EXPRESSION

DEFINED IN BASIC AS: A combination of variables, constants and operators which evaluates to a numeric value.

EXAMPLES:

$(P + 5)/27$
(wherein P has previously been assigned a numeric value.)

$Q - (N + 4)$
(where Q and N have previously been assigned numeric values.)

TERM: ARITHMETIC EVALUATION

DEFINED IN BASIC AS: The process of calculating the value of an expression.

THE ASSIGNMENT OPERATOR

SYMBOL:	=
EXAMPLES:	<pre> 10 LET A = B2 = C = 0 20 LET A9 = C5 30 LET Y = (N-(R+5))/T 40 LET N5 = A + B2 50 LET P5 = P6=P7=A=B=98.6 </pre>
GENERAL FORM:	<u>LET</u> variable = expression

PURPOSE

Assigns an arithmetic or logical value to a variable.

COMMENTS

When used as an assignment operator, = is read "takes the value of," rather than "equals". It is, therefore, possible to use assignment statements such as:

LET X = X+2

This is interpreted by BASIC as: "LET X take the value of (the present value of) X, plus two."

Several assignments may be made in the same statement, as in statements 10 and 50 above.

See Section V, "Logical Operations" for a description of logical assignments.

RELATIONAL OPERATORS

SYMBOLS:	= # <> > < >= <=
EXAMPLES:	<pre> 100 IF A=B THEN 900 110 IF A+B >C THEN 910 120 IF A+B < C+E THEN 920 130 IF C>=D*E THEN 930 140 IF C9<= G*H THEN 940 150 IF P2#C9 THEN 950 160 IF J <> K THEN 950 </pre>

PURPOSE

Determines the logical relationship between two expressions, as

equality: =
 inequality: # or <>
 greater than: >
 less than: <
 greater than or equal to: >=
 less than or equal to: <=

COMMENTS

NOTE: It is not necessary for the novice to understand the nature of logical evaluation of relational operators, at this point. The comments below are for the experienced programmer.

Expressions using relational operators are logically evaluated, and assigned a value of "true" or "false" (the numeric value is 1 for "true," and 0 for false).

When the = symbol is used in such a way that it might have either an assignment or a relational function, BASIC assumes it is an assignment operator. For a description of the assignment statement using logical operators, see Section V, "Logical Operations."

50

1,444,141

50

ARITHMETIC OPERATORS

SYMBOLS:

↑ * / + -

EXAMPLES:

40 LET N1 = X-5
50 LET C2 = N13
60 LET A = (B-C)/4
70 LET X = ((P12)-(Y*X))/N+Q

PURPOSE

Represents an arithmetic operation, as:

exponentiate: ↑
multiply: *
divide: /
add: +
subtract: -

COMMENTS

The “-” symbol is also used as a sign for negative numbers. It is good practice to group arithmetic operations with parentheses when unsure of the exact order of precedence. The order of precedence (hierarchy) is:

↑
* /
+ -

with ↑ having the highest priority. Operators on the same level of priority are acted upon from left to right in a statement. See “Order of Precedence” in this Section for examples.

The symbols + and - are also used to indicate unary plus and unary minus. For example, negative numbers may be expressed in a statement without using parenthesis:

10 LET A1 = -B
20 LET C2 = D ++E
30 LET B5 = B --C

See “Order of Precedence” in this section for examples of how unary + and unary - are interpreted.

THE AND OPERATOR

SYMBOL:	AND
EXAMPLES:	<pre> 60 IF A9<B1 AND C#5 THEN 100 70 IF T7#T AND J=27 THEN 150 80 IF P1 AND R>1 AND N AND V2 THEN 10 90 PRINT X AND Y </pre>

PURPOSE

Forms a logical conjunction between two expressions. If both are "true," the conjunction is "true"; if one or both are "false," the conjunction is "false."

NOTE: It is not necessary for the novice to understand how this operator works. The comments below are for experienced programmers.

COMMENTS

The numeric value of "true" is 1, of "false" is 0.

All non-zero values are "true." For example, statement 90 would print either a 0 or a 1 (the logical value of the expression X AND Y) rather than the actual numeric values of X and Y.

Control is transferred in an IF statement using AND, only when all parts of the AND conjunction are "true." For instance, example statement 80 requires four "true" conditions before control is transferred to statement 10.

See Section V, "Logical Operations" for a more complete description of logical evaluation.

THE OR OPERATOR

SYMBOL:	OR
EXAMPLES:	<pre> 100 IF A>1 OR B<5 THEN 500 110 PRINT C OR D 120 LET D = X OR Y 130 IF (X AND Y) OR (P AND Q) THEN 600 </pre>

PURPOSE

Forms the logical disjunction of two expressions. If either or both of the expressions are true, the OR disjunction is "true"; if both expressions are "false," the OR disjunction is "false."

NOTE: It is not necessary for the novice to understand how this operator works. The comments below are for experienced programmers.

COMMENTS

The numeric values are: "true" = 1, "false" = 0.

All non-zero values are true; all zero values are false.

Control is transferred in an IF statement using OR, when either or both of the two expressions evaluate to "true."

See Section V, "Logical Operations" for a more complete description of logical evaluation.

THE NOT OPERATOR

SYMBOL:

NOT

EXAMPLES:

```

30 LET X = Y = 0
35 IF NOT A THEN 300
45 IF (NOT C) AND A THEN 400
55 LET B5 = NOT P
65 PRINT NOT (X AND Y)
70 IF NOT (A=B) THEN 500

```

PURPOSE

Logically evaluates the complement of a given expression.

NOTE: It is not necessary for the novice to understand how this operator works. The comments below are intended for experienced programmers.

COMMENTS

If $A = 0$, then $\text{NOT } A = 1$; if A has a non-zero value, $\text{NOT } A = 0$.

The numeric values are: "true" = 1, "false" = 0; for example, statement 65 above would print "1", since the expression $\text{NOT } (X \text{ AND } Y)$ is true.

Note that the logical specifications of an expression may be changed by evaluating the complement. In statement 35 above, if A equals zero, the evaluation would be "true" (1); since A has a numeric value of 0, it has a logical value of "false," making $\text{NOT } A$ "true."

See Section V, "Logical Operations" for a more complete description of logical evaluation.

ORDER OF PRECEDENCE

The order of performing operations is:

↑	NOT	unary +	unary -	highest precedence
	*	/		
	+	-		
	Relational Operators			
	AND			
	OR			lowest precedence

COMMENTS

If two operators are on the same level, the order of execution is left to right, for example:

$5 + 6 * 7$	is evaluated as: $5 + (6 * 7)$
$7 / 14 * 2 / 5$	is evaluated as: $(7 / 14) * 2 / 5$

Parentheses override the order of precedence in all cases, for example:

$5 + (6 * 3)$	is evaluated as: $5 + 18$
and	
$3 + (6 + (2 * 2))$	is evaluated as: $3 + (6 + 4)$

Unary + and - may be used; the parentheses are assumed by BASIC. For example:

$A + + B$	is interpreted: $A + (+B)$
$C - + D - 5$	is interpreted: $C - (+D) - 5$

Leading unary + signs are omitted from output by BASIC, but remain in program listings.

STATEMENTS

Statements are instructions to the calculator. They are contained in numbered lines within a program, and execute in the order of their line numbers. Statements cannot be executed without running a program. They tell the calculator what to do while a program is running.

Here are some examples mentioned in Section 1:

LET
PRINT
INPUT

Do not attempt to memorize every detail in the "Statements" subsection; there is too much material to master in a single session. By experimenting with the sample programs and attempting to write your own programs, you will learn more quickly than by memorizing.

THE LET STATEMENT

EXAMPLES:

10 LET A = 5.02
20 LET X = Y7 = Z = 0
30 LET B9 = 5* (X+2)
40 LET D = (3*C2+N)/(A*(N/2))

GENERAL FORM:

statement number LET variable = number or expression or variable ...

PURPOSE

Used to assign or specify the value of a variable. The value may be an expression, a number, or a variable.

COMMENTS

The assignment statement must contain:

1. A statement number,
2. LET is optional
3. The variable to be assigned a value (for example, B9 in statment 30 above),
4. The assignment operator, an = sign,
5. The number, expression or variable to be assigned to the variable (for example, 5*(X+2) in statement 30 above).

Statement 20 in the example above shows the use of an assignment to give the same value (0) to several variables. This is a useful feature for initializing variables in the beginning of a program.

REM

EXAMPLES:

```

10 REM—THIS IS AN EXAMPLE
20 REM: OF REM STATEMENTS
30 REM———///******!!!!
40 REM. STATEMENTS ARE NOT
   EXECUTED BY BASIC

```

GENERAL FORM:

statement number REM any remark or series of characters

PURPOSE

Allows insertion of a line of remarks or comment in the listing of a program.

COMMENTS

Must be preceded by a line number. Any series of characters may follow REM.

REM lines are part of a BASIC program and are printed when the program is listed or punched; however, they are ignored when the program is executing.

Remarks are easier to read if REM is followed by a punctuation mark, as in the example statements.

PRINT

This sample program gives a variety of examples of the PRINT statement. The results are shown below.

```

10 LET A=B=C=10
20 LET D1=E9=20
30 PRINT A,B,C,D1,E9
40 PRINT A/B,B/C/D1+E9
50 PRINT "NOTE THE POWER TO EVALUATE AN
   EXPRESSION AND PRINT THE"
60 PRINT "VALUE IN THE SAME STATEMENT."
70 PRINT
80 PRINT
90 REM* "PRINT" WITH NO OPERAND CAUSES THE
   TELEPRINTER TO SKIP A LINE.
100 PRINT "'A' DIVIDED BY 'E9' =";A/E9
110 PRINT "11111", "22222", "33333", "44444", "55555", "66666"
120 PRINT "11111", "22222", "33333", "44444", "55555", "66666"
130 END

```

-----RESULTS-----

RUN

```

10          10          10          20          20
1          20.05

```

NOTE THE POWER TO EVALUATE AN EXPRESSION AND PRINT THE VALUE IN THE SAME STATEMENT.

'A' DIVIDED BY 'E9' = .5

```

11111      22222      33333      44444      55555
66666
111112222233333444445555566666

```

NOTE: The "," and ";" used in statements 110 and 120 have very different effects on the format.

PRINT, Continued

GENERAL FORM:

statement number PRINT expression , expression , ...
or
statement number PRINT "any text" ; expression ; ...
or
statement number PRINT "text" ; expression ; "text" , "text" , ...
or
statement number PRINT any combination of text and/or expressions
or
statement number PRINT

PURPOSE

Causes the expressions or "text" to be output to the PRINTER.
Causes the printer to skip a line when used without an operand.

COMMENTS

Note the effects of , and ; on the output of the sample program. If a comma is used to separate PRINT operands, five fields are printed per printer line. If semicolon is used, up to twelve "packed" numeric fields are output per printer line (72 characters).

Text in quotes is printed literally.
NOTE: A variable name is considered as a simple expression by BASIC. For example, a statement for the first general form shown above might be:

100 PRINT A1, B2, C3
or
110 PRINT A, Z, X, T9

where the variables represent numeric expressions.
Remember that variable values must be defined in an assignment, INPUT, READ or FOR statement before being used in a PRINT statement.

Ending a PRINT statement with a semicolon causes the output to be printed on the same line, rather than generating a return linefeed after the statement is executed. For example, the sequence:

20 LET X = 1
30 PRINT X;
40 LET X=X+1
50 GO TO 30
:

produces output in this format:

1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24

Similarly, ending a PRINT statement with a comma causes output to fill all five fields on a line before moving to the next line. The trailing comma in statement 30 in the sequence:

20 LET X = 1
30 PRINT X,
40 LET X=X+1
50 GO TO 30
:

produces output in this format:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

A PRINT statement without an operand (statements 70 and 80 in the sample program) generates a return linefeed.

GO TO AND MULTIBRANCH GO TO

EXAMPLES:

10 LET X = 20

40 GO TO X+Y OF 410,420,430

50 GOTO 100

80 GOTO 10

90 GO TO N OF 100,150,180,190

GENERAL FORM:

statement number GO TO *statement number**statement number* GO TO *expression* OF *sequence of statement numbers*PURPOSE

GO TO transfers control to the statement specified.
 GO TO *expression*... round the *expression* to an integer *n* and transfers control to the *n*th statement number following OF.

COMMENTS

GO TO may be written: GOTO or GO TO.

Must be followed by the statement number to which control is transferred, or *expression* OF, and a sequence of statement numbers.GO TO overrides the normal execution sequence of statements in a program. If there is no statement number corresponding to the value of the *expression*, the GO TO is ignored.

Useful for repeating a task infinitely, or "jumping" (GOing TO) another part of a program if certain conditions are present.

GO TO should not be used to enter FOR-NEXT loops; doing so may produce unpredictable results or fatal errors.

GOSUB...RETURN

EXAMPLE:

```
50 READ A2
60 IF A2<100 THEN 80
70 GOSUB 400

380 STOP (STOP frequently precedes the
          first statement of a subroutine,
          to prevent accidental entry.)
390 REM—THIS SUBROUTINE ASKS
    FOR A 1 OR 0 REPLY.
400 PRINT "A2 IS>100"
410 PRINT "DO YOU WANT TO
    CONTINUE?"
420 INPUT N
430 IF N #0 THEN 450
440 LET A2=0
450 RETURN

600 END
```

GENERAL FORM:

statement number GOSUB statement
number starting subroutine

statement number RETURN

PURPOSE

GOSUB transfers control to the specified statement number.
RETURN transfers control to the statement following the GOSUB statement
which transferred control.
GOSUB...RETURN eliminates the need to repeat frequently used groups of
statements in a program.

MULTIBRANCH GOSUB

EXAMPLES:

20 GOSUB 3 OF 100,200,300,400,
500
60 GOSUB N+1 OF 200,210,220
70 GOSUB N OF 80,180,280,380,
480,580

GENERAL FORM:

statement number **GOSUB** expression **OF** sequence of statement numbers...

PURPOSE

GOSUB *expression* rounds the *expression* to an integer *n* and transfeeds control to the *n*th statement number following OF.

COMMENTS

Subroutines should be exited only with a RETURN statement.

The *expression* indicates which of the specified subroutines will be executed. For example, statement 20, above transfers control to the subroutine beginning with statement 300. The *expression* specifies which statement in the sequence of five statements is used as the starting one in the subroutine.

The *expression* is evaluated as an integer. Non-integer values are rounded to the nearest integer.

If the *expression* evaluates to a number greater than the number of statements specified, or less than 1, the GOSUB is ignored.

Statement numbers in the sequence following OF must be separated by commas.

IF... THEN

SAMPLE PROGRAM:

```
10 LET N = 10
20 READ X
30 IF X <= N THEN 60
40 PRINT "X IS OVER"; N
50 GO TO 100
60 PRINT "X IS LESS THAN OR
   EQUAL TO"; N
70 GO TO 20
80 STOP
```

GENERAL FORM:

statement number **IF** *expression relational op expression* **THEN** *statement number*

PURPOSE

Transfers control to a specified statement if a specified condition is true.

COMMENTS

Sometimes described as a conditional transfer; "GO TO" is implied by IF... THEN, if the condition is true. In the example above, if $X \leq 10$, the message in statement 60 is printed (statement 60 is executed).

Since numbers are not always represented exactly in the computer, the = operator should be used carefully in IF... THEN statements. Limits, such as $<=$, $>=$, etc. should be used in an IF expression, rather than =, whenever possible.

If the specified condition for transfer is not true, the program will continue executing in sequence. In the example above, if $X > 10$, the message in statement 40 prints.

The relational operator is optional in logical evaluations.

See Section V, "Logical Operations," for a more complete description of logical evaluation.

FOR...NEXT

EXAMPLES:

```
100 FOR P1 = 1 TO 5
110 FOR Q1 = N TO X
120 FOR R2 = N TO X STEP 2.5
130 FOR S = 1 TO X STEP '
140 NEXT S
150 NEXT R2
160 NEXT Q1
170 NEXT P1
```

Sample Program — Variable Number Of Loops

```
40 PRINT "HOW MANY TIMES DO YOU WANT TO LOOP";
50 INPUT A
60 FOR J = 1 TO A
70 PRINT "THIS IS LOOP"; J
80 READ N1, N2, N3
90 PRINT "THESE DATA ITEMS WERE READ:" N1; N2; N3
100 PRINT "SUM ="; (N1+N2+N3)
110 NEXT J
120 DATA 5, 6, 7, 8, 9, 10, 11, 12
130 DATA 13, 14, 15, 16, 17, 18, 19, 20, 21
140 DATA 22, 23, 24, 25, 26, 27, 28, 29, 30
150 DATA 31, 32, 33, 34
160 END
```

GENERAL FORM:

statement number FOR simple variable = initial value TO final value
or
statement no. FOR simple var. = initial value TO final value STEP
step value

statement number NEXT simple variable

NOTE: The same simple variable must be used in both the *FOR* and *NEXT* statements of a loop.

PURPOSE

Allows controlled repetition of a group of statements within a program

COMMENTS

Initial value, final value and step value may be any expression.

STEP and *step value* are optional; if no step value is specified, the computer will automatically increment by one each time it executes the loop.

How the loop works:

The simple variable is assigned the value of the *initial value*; the value of the simple variable is increased by 1 (or by the *step value*) each time the loop executes. When the value of the *simple variable* passes the *final value*, control is transferred to the statement following the "NEXT" statement.

The *initial, final, and step values* are all evaluated upon entry to the loop and remain unchanged after entry. For example,

FOR I = 1 TO I + 5

goes from 1 to 6; that is, the *final value* does not "move" as I increases with each pass through the loop.

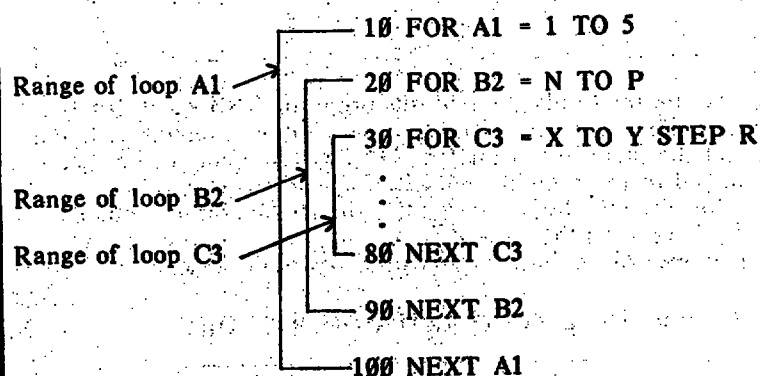
For further details on the STEP feature, see "FOR...NEXT with STEP" in Section III.

Try running the sample program if you are not sure what happens when FOR...NEXT loops are used in a program.

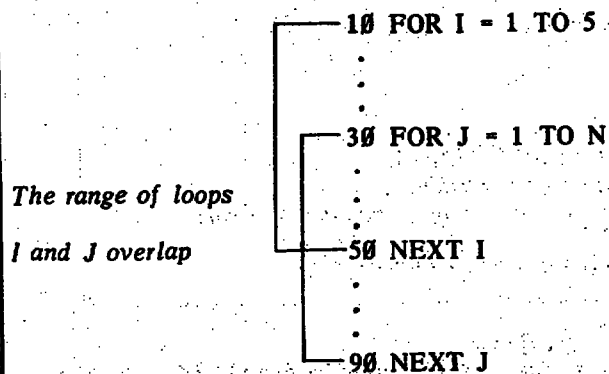
NESTING FOR...NEXT LOOPS

Several FOR...NEXT loops may be used in the same program; they may also be nested (placed inside one another). There are two important features of FOR...NEXT loops:

1. FOR...NEXT loops may be nested.



2. The range of FOR...NEXT loops may not overlap. The loops in the example above are nested correctly. This example shows improper nesting.



READ, DATA AND RESTORE

Sample Program using READ and DATA

```

15 FOR I=1 To 5
20 READ A
40 X=A^2
45 PRINT A; " SQUARED =" ; X
50 NEXT I
55 DATA 5.24,6.75,30.8,72.65,89.72
60 END

```

Each data item may be read only once in this program. TSB keeps track of data with a "pointer." When the first READ statement is encountered, the "pointer" indicates that the first item in the first DATA statement is to be read; the pointer is then moved to the second item of data, and so on.

In this example, after the loop has executed five times, the pointer remains at the end of the data list. To reread the data, it is necessary to reset the pointer. A RESTORE statement moves the pointer back to the first data item.

Sample Program Using READ, DATA and RESTORE

```

20 For I=1 To 5
30 READ A
40 LET X=A^2
50 PRINT A; "SQUARED =" ; X
60 NEXT I
80 RESTORE
100 FOR J=1 To 5
110 READ B
120 LET Y=B^4
130 PRINT B; "TO THE FOURTH POWER =" ; Y
140 NEXT J
150 DATA 5.24,6.75,30.8,72.65,89.72
160 END

```

GENERAL FORM:

Statement number READ variable , variable , ...
 statement number DATA number or string , number or string , ...
 statement number RESTORE
 statement number RESTORE statement number

PURPOSE

The READ statement instructs TSB to read an item from a DATA statement. The DATA statement is used for specifying data in a program. The data is read in sequence from first to last DATA statements, and from left to right within the DATA statement.

The RESTORE statement resets the pointer to the first data item, allowing data to be re-read.

RESTORE followed by a statement number resets the pointer to the first data item, beginning at the specified statement.

COMMENTS

READ statements require at least one DATA statement in the same program. Items in a DATA statement must be separated by commas. String and numeric data may be mixed.

DATA statements may be placed anywhere in a program. The data items will be read in sequence as required.

DATA statements do not execute; they merely specify data.

The RUN command automatically sets the pointer to the first data item.

If you are not sure of the effects of READ, DATA, and RESTORE, try running the sample programs.

WAIT

EXAMPLE:	900 WAIT (1000) 990 WAIT (3000)
GENERAL FORM:	statement number WAIT (expression max. value of 32767)

PURPOSE

Introduces delays into a program. WAIT causes the program to wait the specified number of milliseconds (maximum 32767 milliseconds) before continuing execution.

COMMENTS

The time delay produced by WAIT is not precisely the number of milliseconds specified because there is no provision to account for time elapsed during calculation or terminal-computer communication.
One millisecond = 1/1000 second.

TERM: ROUTINE

DEFINED IN BASIC AS:	A sequence of program statements which produces a certain result.
-----------------------------	---

PURPOSE

Routines are used for frequently performed operations, saving the programmer the work of defining an operation each time he uses it, and saving computer memory space.

COMMENTS

A routine may also be called a program, subroutine, or sub-program. The task performed by a routine is defined by the programmer.

TERM: STRING

DEFINED IN BASIC AS:	0 to 255 printer characters enclosed by quotation marks (one line on a teleprinter terminal).
-----------------------------	---

COMMENTS

Sample strings: "ANY CHARACTERS!*/_—"
"TEXT 1234567..."

Quotation marks may not be used within a string. Strings are used only in PRINT statements.

The statement number PRINT, and quotation marks are not included in the 65 character count. Each statement may contain up to 72 characters. Maximum string length is 72 characters minus 6 characters for "PRINT", two for the quotation marks, and the number of characters in the statement number.

TERM: FUNCTION

DEFINED IN BASIC AS: The mathematical relationship between two variables (X and Y, for example) such that for each value of X there is one and only one value of Y.

COMMENTS

5 The independent variable in a function is called an argument; the dependent variable is the function value. For instance, if X is the argument, the function value is the square root of X, and Y takes the value of the function. 5

TERM: ARRAY OR MATRIX

DEFINED IN BASIC AS: An ordered collection of numeric data (numbers).

COMMENTS

10 Arrays are divided into columns (vertical) and rows (horizontal): 10

C	ROWS
O	
L	
U	
M	
N	
S	

Arrays may have one or two dimensions. For example,

1.0
2.1
3.2
4.3

15 is a one-dimensional array, while 15

6 , 5 , 4
3 , 2 , 1
0 , 9 , 8

20 is a two-dimensional array. 20

25 Array elements are referenced by their row and column position. For instance, if the two examples above were arrays A and Z respectively, 2.1 would be A(2); similarly, 0 would be Z(3,1). The references to array elements are called subscripts, and set apart with parentheses. For example, P(1,5) references the fifth element of the first row of array P; 1 and 5 are the subscripts. In X(M,N) M and N are the subscripts. 25

TERM: WORD

DEFINED IN BASIC AS: The amount of computer memory space occupied by the two teleprinter characters.

COMMENTS

30 Numbers require two words of memory space when stored as numbers. When used within a string, numbers require 1/2 word of space per character in the number. 30

Standard mathematical and trigonometric functions are convenient timesavers for programmers at any level. They are treated as numeric expressions by BASIC.

A negative step size may be used, as shown in statement 40 above.

GENERAL MATHEMATICAL FUNCTIONS

EXAMPLES:

```
642 PRINT EXP(N); ABS(N)
652 IF RND (0) >=.5 THEN 900
662 IF INT (R) # 5 THEN 910
672 PRINT SQR (X); LOG (X)
```

GENERAL FORM:

The general mathematical functions may be used as expressions, or as parts of an expression.

PURPOSE

Facilitates the use of common mathematical functions by pre-defining them as:

ABS (*expression*) the absolute value of the expression;

EXP (*expression*) the constant e raised to the power of the expression value (in statement 642 above, e^N);

INT (*expression*) the largest integer $\leq$ the expression;

LOG (*expression*) the logarithm of the positively valued expression to the base e ;

RND (*expression*) a random number between 1 and 0; the expression is a dummy argument;

SQR (*expression*) the square root of the positively valued expression.

COMMENTS

The RND function is restartable; the sequence of random numbers using RND is identical each time a program is RUN.

TRIGONOMETRIC FUNCTIONS

EXAMPLES:

```
500 PRINT SIN(X); COS(Y)
510 PRINT 3*SIN(B); TAN (C2)
520 PRINT ATN (22.3)
530 IF SIN (A2) < 1 THEN 800
540 IF SIN (B3) = 1 AND SIN(X) < 1
    THEN 90
```

PURPOSE

Facilitates the use of common trigonometric functions by pre-defining the as:

SIN (*expression*) the sine of the expression

COS (*expression*) the cosine of the expression

TAN (*expression*) the tangent of the expression

ATN (*expression*) the arctangent of the expression

COMMENTS

The function is of the value of the expression (the value in parentheses, also called the argument).

The trigonometric functions may be used as expressions or parts of an expression.

The angle of the trigonometric functions can be specified as radians, degrees, or grads by executing a RAD, DEG, or GRAD statement. The calculator assumes radians if not specified.

THE TAB AND SGN FUNCTIONS

EXAMPLES:

```

500 IF SGN (X) # 0 THEN 800
510 LET Y = SGN(X)
520 PRINT TAB (5); A2; TAB (20)
    "TEXT"
530 PRINT TAB (N),X,Y,Z2
540 PRINT TAB (X+2) "HEADING",
    R5

```

GENERAL FORM:

The TAB and SGN may be used as expressions or parts of an expression. The function forms are:

TAB (expression indicating number of spaces to be moved)
SGN (expression)

PURPOSE

TAB (*expression*) is used only in a PRINT statement, and causes the terminal typeface to move to the space number specified by the expression (0 to 71). The *expression* value after TAB is rounded to the nearest integer. Expression values greater than 71 cause a return linefeed to be generated.

SGN (*expression*) returns a 1 if the expression is greater than 0, returns a 0 if the expression equals 0, returns a -1 if the expression is less than 0.

SECTION IV

MATRICES

This section explains matrix manipulation. It is intended to show the matrix capabilities of BASIC and assumes that the programmer has some knowledge of matrix theory.

TERM: MATRIX (ARRAY)

DEFINED IN BASIC AS: An ordered collection of numeric data (numbers).

Matrix elements are referenced by subscripts following the matrix variable, indicating the row and column of the element. For example, if matrix A is:

```

1 2 3
4 5 6
7 8 9

```

(the element 5 is referenced by A(2,2); likewise, 8 is A(3,2).

See Section III, "Vocabulary" for a more complete description of matrices.

DIM

EXAMPLES:

110 DIM A (50), B(20,20)

120 DIM Z (5,20)

130 DIM S (5,25)

140 DIM R (4,4)

GENERAL FORM:

statement number DIM matrix variable (integer) ...

or

statement number DIM matrix variable (integer , integer) ...

PURPOSE

Reserves working space in memory for a matrix.
The maximum integer value (matrix bound) is 255.

COMMENTS

The integers refer to the number of matrix elements if only one dimension is supplied, or to the number of rows and columns respectively, if two dimensions are given.

A matrix (array) variable is any single letter from A to Z.

Arrays not mentioned in a DIM statement are assumed to have 10 elements if one-dimensional, or 10 rows and columns if two-dimensional.

The working size of a matrix may be smaller than its physical size. For example, an array declared 9×9 in a DIM statement may be used to store fewer than 81 elements; the DIM statement supplies only an upper bound on the number of elements.

The absolute maximum matrix size depends on the memory size of the computer.

MAT...ZER

EXAMPLES:

305 MAT A = ZER

310 MAT Z = ZER (N)

315 MAT X = ZER (30, 10)

320 MAT R = ZER (N, P)

GENERAL FORM:

statement number MAT matrix variable = ZER

or

statement number MAT matrix variable = ZER (expression)

or

statement number MAT matrix variable = ZER (expression , expression)

PURPOSE

Sets all elements of the specified matrix equal to 0; a new working size may be established.

COMMENTS

The new working size in a MAT...ZER is an implicit DIM statement, and may not exceed the limit set by the DIM statement on the total number of elements in an array.

Since 0 has a logical value of "false," MAT...ZER is useful in logical initialization.

MAT...CON

EXAMPLES: 205 MAT C = CON
 210 MAT A = CON (N,N)
 220 MAT Z = CON (5,20)
 230 MAY Y = CON (50)

GENERAL FORM:

statement number MAT matrix variable = CON
 or
 statement number MAT matrix variable = CON (expression)
 or
 statement number MAT matrix variable = CON (expression ,
 expression)

PURPOSE

Sets up a matrix with all elements equal to 1; a new working size may be specified, within the limits of the original DIM statement on the total number of elements.

COMMENTS

The new working size (an implicit DIM statement) may be omitted as in example statement 205.

Note that since 1 has a logical value of "true," the MAT...CON statement is useful for logical initialization.

The expressions in new size specifications should evaluate to integers. Non-integers are rounded to the nearest integer value.

PRINTING SINGLE MATRIX ELEMENTS

EXAMPLES: 800 PRINT A(3)
 810 PRINT A(3,3);
 820 PRINT F(X);E; C5;R(N)
 830 PRINT G(X,Y)
 840 PRINT Z(X,Y), Z(1,5), Z(X+N),
 Z(Y+M)

GENERAL FORM:

statement number PRINT matrix variable (expression) ...
 or
 statement number PRINT matrix variable (expression , expression) ...

PURPOSE

Causes the specified matrix element(s) to be printed.

COMMENTS

Expressions used as subscripts should evaluate to integers. Non-integers are rounded to the nearest integer value.

A trailing semicolon packs output into twelve elements per teleprinter line, if possible (statement 810 above). A trailing comma or return prints five elements per line.

Expressions (or subscripts) following the matrix variable designate the row and column of the matrix element. Do not confuse these with new working size specifications, such as those following a MAT IDN statement.

INPUTTING SINGLE MATRIX ELEMENTS

EXAMPLES:

```

600 INPUT A(5)
610 INPUT B(5,8)
620 INPUT R(X), N, A(3,3),S,T
630 INPUT Z(X,Y), P3, W
640 INPUT Z(X,Y), Z(X+1, Y+1),
    Z(X+R3, Y+S2)

```

GENERAL FORM:

statement number INPUT matrix variable (expression)
or
statement number INPUT matrix variable (expression , expression)

PURPOSE

Allows input of a specified matrix element from the keyboard.

COMMENTS

The subscripts (in *expressions*) used after the matrix variable designate the row and column of the matrix element. Do not confuse these expressions with working size specifications, such as those following a MAT READ statement.

Expressions used as subscripts should evaluate to integers. Non-integers are rounded to the nearest integer value.

Inputting, printing, and reading individual array elements are logically equivalent to simple variables and may be intermixed in INPUT, PRINT, and READ statements.

MAT PRINT

EXAMPLES:

```

500 MAT PRINT A
505 MAT PRINT A;
515 MAT PRINT A,B,C
520 MAT PRINT A,B,C;

```

GENERAL FORM:

statement number MAT PRINT matrix variable
or
statement number MAT PRINT matrix variable , matrix variable

PURPOSE

Causes an entire matrix to be printed, row by row, with double spacing between rows.

COMMENTS

Matrices may be printed in "packed" rows up to 12 elements wide by using the ";" separator, as in example statement 505.

READING MATRIX ELEMENTS

EXAMPLES: 900 READ A(6)
 910 READ A(9,9)
 920 READ C(X); P; R7
 930 READ C(X,Y)
 940 READ Z(X,Y), P(R2, S5), X(4)

GENERAL FORM:
statement number **READ** *matrix variable (expression)*
or
statement number **READ** *matrix variable (expression , expression) ...*

PURPOSE

Causes the specified matrix element to be read from the current DATA statement.

COMMENTS

Expressions (used as subscripts) should evaluate to integers. Non-integers are rounded to the nearest integer.

Expressions following the matrix variable designate the row and column of the matrix element. Do not confuse these with working size specifications, such as those following MAT READ statement.

The MAT READ statement is used to read an entire matrix from DATA statements. See details in this section.

MAT READ

EXAMPLES: 350 MAT READ A
 370 MAT READ B(5),C,D
 380 MAT READ Z (5,8)
 390 MAT READ N (P3,Q7)

GENERAL FORM:
statement number **MAT READ** *matrix variable*
or
statement number **MAT READ** *matrix variable (expression) ...*
or
statement number **MAT READ** *matrix variable (expression , expression) ...*

PURPOSE

Reads an entire matrix from DATA statements. A new working size may be specified, within the limits of the original DIM statement.

COMMENTS

MAT READ causes the entire matrix to be filled from the current DATA statement in the row, column order: 1,1; 1,2; 1,3; etc. In this case, the DIM statement controls the number of elements read.

MATRIX ADDITION

EXAMPLES:

310 MAT C = B + A
 320 MAT X = X + Y
 330 MAT P = N + M

GENERAL FORM:

statement number **MAT** *matrix variable* = *matrix variable* + *matrix variable*

PURPOSE

Establishes a matrix equal to the sum of two matrices of identical dimensions; addition is performed element-by-element.

COMMENTS

The resulting matrix must be previously mentioned in a DIM statement if it has more than 10 elements, or 10×10 elements if two-dimensional. Dimensions must be the same as the operand matrices.

The same matrix may appear on both sides of the = sign, as in example statement 320.

MATRIX SUBTRACTION

EXAMPLES:

550 MAT C = A - B
 560 MAT B = B - Z
 570 MAT X = X - A

GENERAL FORM:

statement number **MAT** *matrix variable* = *matrix variable* - *matrix variable*

PURPOSE

Establishes a matrix equal to the difference of two matrices of identical dimensions; subtraction is performed element-by-element.

COMMENTS

The resulting matrix must be previously mentioned in a DIM statement if it has more than 10 elements, or 10×10 elements if two-dimensional. Its dimension must be the same as the operand matrices.

The same matrix may appear on both sides of the = sign, as in example statement 560.

MATRIX MULTIPLICATION

EXAMPLES:

```
930 MAT Z = B * C
940 MAT X = A * A
950 MAT C = Z * B
```

GENERAL FORM:

statement number **MAT** *matrix variable* = *matrix variable* * *matrix variable*

PURPOSE

Establishes a matrix equal to the product of the two specified matrices.

COMMENTS

Following the rules of matrix multiplication, if the dimensions of matrix B = (P,N) and matrix C = (N,Q), multiplying matrix B by matrix C results in a matrix of dimensions (P,Q).

Note that the product matrix must have an appropriate working size.

The same matrix variable may not appear on both sides of the = sign.

SCALAR MULTIPLICATION

EXAMPLES:

```
110 MAT A = (5) * B
115 MAT C = (10) * C
120 MAT C = (N/3) * X
130 MAT P = (Q7*N5) * R
```

GENERAL FORM:

statement number **MAT** *matrix variable* = (*expression*) * *matrix variable*

PURPOSE

Establishes a matrix equal to the product of a matrix multiplied by a specified expression (number); that is, each element of the original matrix is multiplied by the number.

COMMENTS

The resulting matrix must be previously mentioned in a DIM statement if it contains more than 10 elements (10 × 10 if two-dimensional).

The same matrix variable may appear on both sides of the = sign.

Both matrices must have the same working size.

COPYING A MATRIX

EXAMPLES:

405 MAT B = A
 410 MAT X = Y
 420 MAT Z = B

GENERAL FORM:

statement number MAT matrix variable = matrix variable

PURPOSE

Copies a specified matrix into a matrix of the same dimensions; copying is performed element-by-element.

COMMENTS

The resulting matrix must be previously mentioned in a DIM statement if it has more than 10 elements, or 10 × 10 if two-dimensional. It must have the same dimensions as the copied matrix.

IDENTITY MATRIX

EXAMPLES:

205 MAT A = IDN
 210 MAT B = IDN (3,3)
 215 MAT Z = IDN (Q5, Q5)
 220 MAT S = IDN (6, 6)

GENERAL FORM:

statement number MAT array variable = IDN

or

statement number MAT array variable = IDN (expression ,
 expression)

PURPOSE

Establishes an identity matrix (all 0's, with a diagonal from left to right of all 1's); a new working size may be specified.

COMMENTS

The IDN matrix must be two-dimensional and square.
 Specifying a new working size has the effect of a DIM statement.
 Sample identity matrix:

1	0	0
0	1	0
0	0	1

MATRIX TRANSPOSITION

EXAMPLES:

959 MAT Z = TRN (A)
969 MAT X = TRN (B)
979 MAT Z = TRN (C)

GENERAL FORM:

statement number MAT matrix variable = TRN (matrix variable)

PURPOSE

Establishes a matrix as the transposition of a specified matrix (transposes rows and columns).

COMMENTS

Sample transposition:

Original
1 2 3
4 5 6
7 8 9

Transposed
1 4 7
2 5 8
3 6 9

Note that the dimensions of the resulting matrix must be the reverse of the original matrix. For instance, if A has dimensions of 6,5 and MAT C = TRN (A), C must have dimensions of 5,6.

Matrices cannot be transposed or inverted into themselves.

MATRIX INVERSION

EXAMPLES:

380 MAT A = INV(B)
390 MAT C = INV(A)
400 MAT Z = INV(Z)

GENERAL FORM:

statement number MAT matrix variable = INV (matrix variable)

PURPOSE

Establishes a square matrix as the inverse of the specified square matrix of the same dimensions.

COMMENTS

The inverse is the matrix by which you multiply the original matrix to obtain an identity matrix.

For example,

Original
 $\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$

x

Inverse
 $\begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}$

=

Identity
 $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Number representation in BASIC is accurate to 6—7 decimal digits; matrix elements are rounded accordingly.

SECTION V LOGICAL OPERATIONS

LOGICAL VALUES AND NUMERIC VALUES

A distinction should be made between logical values and the numeric values produced by logical evaluation, when using the logical capability of BASIC.

The logical value of an expression is determined by definitions established in the user's program.

The numeric values produced by logical evaluation are assigned by BASIC. The user may not assign these values.

Logical value is the value of an expression or statement, using the criteria:
any nonzero expression value = "true"
any expression value of zero = "false"

When an expression or statement is logically evaluated, it is assigned one of two numeric values, either:

1, meaning the expression or statement is "true",
or
0, meaning the expression or statement is "false".

RELATIONAL OPERATORS

There are two ways to use the relational operators in logical evaluations:
1. As a simple check on the numeric value of an expression.

EXAMPLES:

```
150 IF B=7 THEN 600  
200 IF A#27.65 THEN 700  
300 IF (Z/10)>0 THEN 800
```

When a statement is evaluated, if the "IF" condition is currently true (for example, B = 7 in statement 150), then control is transferred to the specified statement; if it is not true, control passes to the next statement in the program.

Note that the numeric value produced by the logical evaluation is unimportant when the relational operators are used in this way. The user is concerned only with the presence or absence of the condition indicated in the IF statement.

2. As a check on the numeric value produced by logically evaluating an expression, that is: "true" = 1, "false" = 0.

EXAMPLES:

```
610 LET X=27  
615 PRINT X=27  
620 PRINT X#27  
630 PRINT X>=27
```

The example PRINT statements give the numeric values produced by logical evaluation. For instance, statement 615 is interpreted by BASIC as "Print 1 if X equals 27, 0 if X does not equal 27." There are only two logical alternatives; 1 is used to represent "true," and 0 "false."

The numeric value of the logical evaluation is dependent on, but distinct from, the value of the expression. In the example above, X equals 27, but the numeric value of the logical expression X=27 is 1 since it describes a "true" condition.

BOOLEAN OPERATORS

There are two ways to use the Boolean Operators.

1. As logical checks on the value of an expression or expressions.

EXAMPLES:

```
510 IF A1 OR B THEN 670
520 IF B3 AND C9 THEN 680
530 IF NOT C9 THEN 690
540 IF X THEN 700
```

Statement 510 is interpreted: "If either A1 is true (has a non-zero value) or B is true (has a non-zero value), then transfer control to statement 670."

Similarly, statement 540 is interpreted: "If X is true (has a non-zero value), then transfer control to statement 700."

The Boolean operators evaluate expressions for their logical values only: these are "true" = any non-zero value, "false" = zero. For example, if B3 = 9 and C9 = -5, statement 520 would evaluate to "true," since both B3 and C9 have a non-zero value.

2. As a check on the numeric value produced by logically evaluating an expression, that is: "true" = 1, "false" = 0.

EXAMPLES:

```
490 LET B = C = 7
500 PRINT B AND C
510 PRINT C OR B
520 PRINT NOT B
```

Statements 500—520 return a numeric value of either 1, indicating that the statement has a logical value of "true", or 0, indicating a logical value of "false".

Note that the criteria for determining the logical values are:

true = any non-zero expression value
false = an expression value of 0.

The numeric value 1 or 0 is assigned accordingly.

SECTION VI SYNTAX REQUIREMENTS OF BASIC

LEGEND

::= "is defined as..."

| "or"

< > enclose an element of BASIC

LANGUAGE RULES

1. The <com statement>, if any exists, must be the first statement presented and have the lowest sequence number; the last statement must be an <END statement>.
2. A sequence number may not exceed 9999 and must be non-zero.
3. Exponent integers may not have more than two digits.
4. A formal bound may not exceed 255 and must be non-zero.
5. A subroutine number must lie between 1 and 63, inclusive.
6. Strings may not contain the quote character (").
7. A <bound part> for an IDN must be doubly subscripted.
8. An array may not be inverted or transposed into itself.
9. An array may not be replaced by itself multiplied by another array.

SYNTAX REQUIREMENTS

<basic program>	::= <program statement> <basic program> <program statement> ⁽¹⁾
<program statement>	::= <sequence number> <basic statement> carriage return
<sequence number>	::= <integer> ⁽²⁾
<basic statement>	::= <let statement> <dim statement> <com statement> <def statement> <rem statement> <go to statement> <if statement> <for statement> <next statement> <gosub statement> <return statement> <end statement> <stop statement> <wait statement> <call statement> <data statement> <read statement> <restore statement> <input statement> <print statement> <mat statement>
<let statement>	::= <let head> <formula>
<let head>	::= LET <variable> = <let head> <variable> =
<formula>	::= <conjunction> <formula a> OR <conjunction>
<conjunction>	::= <boolean primary> <conjunction> AND <boolean primary>
<boolean primary>	::= <arithmetic expression> <boolean primary> <relational operator> <arithmetic expression>
<arithmetic expression>	::= <term> <arithmetic expression> + <term> <arithmetic expression> - <term>
<term>	::= <factor> <term> * <factor> <term> / <factor>
<factor>	::= <primary> <sign> <primary> NOT <primary>
<primary>	::= <operand> <primary> ↑ <operand>
<relational operator>	::= > < = <= ≠ <
<operand>	::= <variable> <unsigned number> <system function> <function> <formula operand>
<variable>	::= <simple variable> <subscripted variable>
<simple variable>	::= <letter> <letter> <digit>
<subscripted variable>	::= <array identifier> <subscript head> <subscript> <right bracket>
<array identifier>	::= <letter>
<subscript head>	::= <left bracket> <left bracket> <subscript>
<subscript>	::= <formula>
<letter>	::= A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
<digit>	::= 0 1 2 3 4 5 6 7 8 9
<left bracket>	::= ([
<right bracket>	::=)]
<sign>	::= + -
<unsigned number>	::= <decimal part> <decimal part> <exponent>

SYNTAX REQUIREMENTS, Continued

<decimal part>	::= <integer> <integer>.<integer> <integer>
<integer>	::= <digit> <integer><digit>
<exponent>	::= E<integer> E<sign><integer> ⁽³⁾
<system function>	::= <system function name><parameter part>
<system function name>	::= SIN COS TAN ATN EXP LOG ABS SQR INT RND SGN
<parameter part>	::= <left bracket><actual parameter><right bracket>
<actual parameter>	::= <formula>
<function>	::= FN<letter><parameter part>
<formula operand>	::= <left bracket><formula><right bracket>
<dim statement>	::= DIM<formal array list>
<formal array list>	::= <formal array> <formal array list>,<formal array>
<formal array>	::= <array identifier><formal bound head><formal bound> <right bracket>
<formal bound head>	::= <left bracket> <left bracket><formal bound>
<formal bound>	::= <integer> ⁽⁴⁾
<com statement>	::= COM<formal array list>
<def statement>	::= DEF FN<letter><left bracket><formal parameter> <right bracket>=<formula>
<formal parameter>	::= <simple variable>
<rem statement>	::= REM<character string>
<character string>	::= any teletype character except carriage return, alt mode, escape, rubout, or line feed, or null, control B, control C, left arrow
<goto statement>	::= GO TO<sequence number>
<if statement>	::= IF<formula>THEN<sequence number>
<for statement>	::= <for head> <for head>STEP<step size>
<for head>	::= FOR<for variable>=<initial value>TO<limit value>
<for variable>	::= <simple variable>
<initial value>	::= <formula>
<limit value>	::= <formula>
<step size>	::= <formula>
<next statement>	::= NEXT<for variable>
<gosub statement>	::= GOSUB<sequence number>
<return statement>	::= RETURN

SYNTAX REQUIREMENTS, Continued

<end statement>	::= END
<stop statement>	::= STOP
<wait statement>	::= WAIT <parameter part>
<call statement>	::= CALL <call head> <right bracket>
<call head>	::= <left bracket> <subroutine number> <call head> , <actual parameter>
<subroutine number>	::= <integer> ⁽⁹⁾
<data statement>	::= DATA <constant> <data statement> , <constant>
<constant>	::= <unsigned number> <sign> <unsigned number>
<read statement>	::= READ <variable list>
<variable list>	::= <variable> <variable list> ' <variable>
<restore statement>	::= RESTORE
<input statement>	::= INPUT <variable list>
<print statement>	::= <print head> <print head> <print formula>
<print head>	::= PRINT <print head> <print part>
<print part>	::= <string> <string> <delimiter> <print formula> <delimiter> <print formula> <string> <print formula> <string> <delimiter>
<string>	::= "<character string>" ⁽⁶⁾
<delimiter>	::= , ;
<print formula>	::= <formula> TAB <parameter part>
<mat statement>	::= MAT <mat body>
<mat body>	::= <mat read> <mat print> <mat replacement>
<mat read>	::= READ <actual array> <mat read> , <actual array>
<actual array>	::= <array identifier> <array identifier> <bound part>
<bound part>	::= <actual bound head> <actual bound> <right bracket>
<actual bound head>	::= <left bracket> <left bracket> <actual bound> ,
<actual bound>	::= <formula>
<mat print>	::= PRINT <mat print part> PRINT <mat print part> <delimiter>
<mat print part>	::= <array identifier> <mat print part> <delimiter> <array identifier>
<mat replacement>	::= <array identifier> = <mat formula>
<mat formula>	::= <array identifier> <mat function> <array identifier> <mat operator> <array identifier> <formula operand> * <array identifier>

SYNTAX REQUIREMENTS, Continued

<mat function> ::= <mat initialization>|<mat initialization> <bound part>|
INV<array parameter>|TRN<array parameter>
<mat initialization> ::= ZER|CON|IDN ⁽⁷⁾
<array parameter> ::= <left bracket> <array identifier> <right bracket> ⁽⁸⁾
<mat operator> ::= +|-|* ⁽⁹⁾

STRINGS PLUG-IN READ-ONLY MEMORY MODULE

The strings plug-in read-only memory module makes available to the user all of the string variable functions and operations associated with standard BASIC programming language. Two additional functions not usually provided in most versions of BASIC language have been implemented. These include a POS function for determining the position of a substring within a string and a VAL function for determining the numeric value of a string. A discussion of these and other string functions and operations follows.

String
A set of 1 to 255 characters or the null string (no characters).
e.g. "ABCDEF"
"12345"
""

String Variable
A variable used to store strings; consists of a single letter (A to Z) followed by a \$.
e.g. A\$; B\$; Z\$

Substring Variable
A single character or a set of contiguous characters from within a string variable. The substring is defined by a subscripted string variable. A single subscript specifies the first character of the substring and implies that all characters following are part of the substring. Two subscripts specify the first and last characters of the substring.
e.g. A\$ = "ABCDEF"
A\$(4) = DEF
A\$(1, 3) = ABC

DIM Statement
General Form:
stmt # DIM string var (#chars in string)
Purpose:
Reserves storage space for strings longer than 1 character.

Comments:
The number of characters specified for a string in its DIM statement must be expressed as an integer from 1 to 255. Strings not mentioned in a DIM statement are assumed to have length 1. The length mentioned in the DIM statement specifies the maximum number of characters which may be assigned and is known as the physical length. The actual length is the actual number of characters which has been assigned.
e.g. DIM A\$(10), B(5, 5), B\$(255)

Source String Semantic Description
A source string is an entity from which a string value is extracted. If no substring designator is specified for a string variable, the value of the source string is the entire logical string currently assigned to it. Substring designator expressions must be at least 1 in value and the second may be no smaller than one less than the first. If one subscript is given, the value is the entire logical string beginning at the

character specified by the subscript. The subscript may be no larger than the actual length of the string plus 1. If two substring expressions are given, the source string value is the substring whose first and last characters are designated. If the specified substring extends beyond the logical length of the string, spaces are used to fill out the substring.

Assignment Statement

General Form:

line # [LET] destination string = source string

Destination String:

A destination string is an entire string variable or part of a string variable into which a source string is to be copied. The definition of the action depends on the number of substring subscripts specified in the destination string.

If no subscript qualifier is specified, the entire destination variable is replaced by the source string. The physical length of the destination string must be large enough to accommodate the entire source string.

If one substring subscript is specified, the entire value of the destination variable, beginning at the designated character, is replaced by the source string. That part of the destination variable preceding the subscript value is unchanged. The subscript value must be no more than one greater than the actual length of the destination variable.

If two subscripts are specified, the first substring subscript must be no more than one greater than the logical length of the destination variable, and the second subscript must be no greater than the physical length of the destination variable. The specified section of the destination string is replaced by the source string. If the source string is longer than the destination, it is truncated on the right. If the source string is shorter, as many trailing blanks are appended as necessary. The new actual length of the destination string is the larger of the old actual length and the second subscript.

String Input Statement

General Form:

stmt # INPUT string or substring variable

Purpose:

Allows string values to be entered from the keyboard.

Comments:

Numeric variables may be used in the same input statement as string variables. Placing a single string variable in an input statement allows the string value to be entered without enclosing it in quotation marks. If multiple string variables are used each string value must be enclosed in quotation marks, and the values separated by commas.

e.g. 10 INPUT A\$, B\$, C, A1, A\$(1, 5)

String Print Statement

General Form:

stmt # PRINT string or substring variable, —

Purpose:

Causes the current value of the specified string or substring variable to be output on the standard output device.

Comments:

Strings and numeric values may be mixed in a print statement. String variables are specified identically to numeric variables. They are printed under the same format rules as quote fields. String and substring variables are printed as source strings.

A maximum of 72 characters can be printed using the print statement, i.e., all strings >72 characters are truncated at 72 characters.

String Read Statement

General Form:

stmt # READ string or substring variable

Purpose:

Causes the value of a specified string or substring variable to be read from a data statement.

Comments:

Mixed string and numeric values may be read. If the wrong data type is given in the data statement an error is given.

e.g. 10 READ A, B\$(1, 10), C\$, B

String If Statement

General Form:

stmt # IF str.var. rel.oper. string THEN stmt #

Purpose:

Compares two strings. If the specified condition is true, control is transferred to the specified statement.

Comments:

Strings are compared one character at a time, from left to right; the first difference determines the relation. If one string ends before a difference is found, the shortest string is considered the smaller one.

Characters are compared by their ASCII representations.

The relational operators allowed are: =, ≠, <, >, <=, >=, <>

e.g. 10 IF A\$ = "SAM" THEN s0

String Data Statement

General Form:

stmt # DATA "string text", "string text", —

Purpose:

Specifies data in a program (string or/and numeric)

Comments:

String values must be enclosed by quotation marks and separated by commas.

String and numeric values may be mixed in a single data statement.

e.g. 10 DATA "ABC", 1.2, "DEF"

String Write Statement

General Form:

stmt # WRITE (device # , format stat #)
string or substring variable

Purpose:

Similar to the print statement but allows the specification of the output device and format statement. No field specifications are made for string variables in the format statement. They are treated identically to string constants (quote fields).

String Display Statement

General Form:

stmt # DISP string or substring variable

Purpose:

Identical to print statement except output device is 32 character display

LEN Function

General Form:

LEN string or LEN (string)

Purpose:

Obtain the length of a string for use in an arithmetic expression.

Comment:

The actual length is found which is not necessarily the same as the physical length reserved in the DIM statement.

e.g. LEN ("ABCD") = 4

POS Function

General Form:

POS string , string or POS (string , string)

Purpose:

Determine the position of a substring within a string.

Comments:

If the second string argument is a part of the first, the value of the function is the position in the first string at which the second string starts. If the second string in the argument is not a part of the first, the value of the function is zero.

e.g. POS ("ABCD" , "C") = 3

BNF SYNTAX DESCRIPTION — Continued

	<input statement>	::= <u>INPUT</u> <variable list>	
	<print statement>	::=<print 1> <print 2>	
	<print 1>	::= <u>PRINT</u> <print 2>, <print 2>; <print 3>	
5	<print 2>	::=<print 1> <print expression> <print 3>	5
	<print 3>	::=(type statement) <literal string>	
	<print expression>	::=<expression> <source string>	
10	WRITE and DISP statements have the same syntax as the PRINT statement.		10
	<LEN function>	::= <u>LEN</u> <source string> <u>LEN</u> (<source string>)	
	<VAL function>	::= <u>VAL</u> <source string> <u>VAL</u> (<source string>)	
15	<POS function>	::= <u>POS</u> <source string>,<source string> <u>POS</u> (<source string>,<source string>)	15

EXTENDED INPUT/OUTPUT PLUG-IN READ-ONLY MEMORY MODULE

The extended I/O read-only memory module (hereinafter referred to as the extended I/O ROM) provides additional functions and statements so that the calculator can be made compatible with a wide variety of peripheral devices. Added functions include decimal-to-binary and octal-to-decimal conversion, status code inquiry for peripheral devices, control of spacing and line feeds in output records, and others. Use of these functions requires no special programming techniques; once the ROM is plugged in, its functions and statements become a part of the calculator, in the same way as, for example, the square root function is part of the calculator.

Some read-only memory units decrease the amount of programmable memory available to the user by automatically requiring a portion of that memory for their own internal usage, but the extended I/O ROM has no such requirement and does not affect memory availability.

The following table describes the extended I/O ROM functions and statements. Parameters shown underlined in the table are explained immediately following the table. Parameters shown in brackets may or may not be included as parts of a statement.

FUNCTION MNEMONIC	DESCRIPTION OF FUNCTION	SYNTAX
BIN	Converts decimal expression to its binary equivalent. For use in output-to-binary storage device; also provides increased control of print format.	BIN <u>exp</u>
OCT	Converts octal expression to decimal equivalent. For use in construction of code conversion tables; see "Conversion Tables".	OCT <u>exp</u>
STAT	Returns code of operational status (on, off, wait, etc.) for the device specified by the select code.	STAT <u>select code</u>
CHAR	Returns one byte of data from the device specified by the select code regardless of the data structure.	CHAR <u>select code</u>
LIN	Advances printer or typewriter the number of lines represented by the expression.	LIN <u>exp</u>
SPA	Advances printer or typewriter carriage the number of spaces represented by the expression.	SPA <u>exp</u>
ROT	Converts expression 1 to binary equivalent; performs rotation right the number of positions represented by expression 2; returns decimal equivalent. For use in special input or output code translation.	ROT (<u>exp 1, exp 2</u>)
INOR	Combines binary equivalents of expression 1 and expression 2 in an "inclusive or" logic operation. Returns decimal equivalent. For use in special input or output code translation.	INOR (<u>exp 1, exp 2</u>)

99

FUNCTION MNEMONIC	DESCRIPTION OF FUNCTION	SYNTAX
BIAND	Combines binary equivalents of expression 1 and expression 2 in an "and" logic operation. Returns decimal equivalent. For use in special input or output translation.	BIAND (exp 1, exp 2)

STATEMENT SYNTAX	DESCRIPTION OF STATEMENT
ENTER	(select code or string name, format[, conversion table]) list[, FOR function] Inputs data from named device or string with optional conversion to ASCII code; includes capability for iteration.
OUTPUT	(select code or string name, format[, conversion table]) list Outputs data to named device or string with optional conversion to ASCII code.

	PARAMETER	EXPLANATION	
	<u>exp</u>	Expression	
5	<u>select code</u>	A numeric code, from 1 to 15, uniquely representing the input or output device, as follows: select code 1—9 User assignable. select code 10 Cassette Memory. select code 11—13 reserved. select code 14 Plotter. select code 15 Typewriter or Printer.	5
10	<u>string name</u>	A single letter followed by a "\$". Valid only when String ROM is also plugged in.	10
15	<u>format</u>	To reference a FORMAT statement, the line number of that statement is shown; for free-form data, an asterisk (*) is shown.	15
	<u>conversion table</u>	The variable or array name given to a conversion table. See heading "Conversion Tables".	
20	<u>list</u>	A list of variables, literals, expressions or numerics separated by commas.	20
	<u>FOR functions</u>	To input multiple data items from one record into an array. Syntax as follows: <div style="text-align: right;">array</div> (FOR var 1 = exp TO exp, [(FOR var 2 = exp TO exp,] name (var 1[, var 2)]))	

100

CONVERSION TABLES

Using a pre-established conversion table, a string of characters or an array of data can be converted from one code to another. The calculator makes use of standard ASCII* codes. Let us refer to all non-ASCII representation codes used by printers, card readers, paper tape readers, punches, typewriters, etc. as "foreign codes."

A conversion table is defined in the BASIC language program DIM statement. Only single-dimensioned integer arrays are considered valid for use as conversion tables. In the following DIM statements, A and B are valid array structures for conversion tables, but C, D, and E are not.

DIM AI (150), CI(20, 30)

DIM BI (80), D(200), E(15, 15)

In order to insert conversion table information in the array, the foreign code must first be known. Suppose your paper tape reader uses EIA** coded tape. The chart following shows symbols and the equivalent tape punches for EIA code, with the octal code equivalent for each symbol.

- * American Standard Code for Information Interchange.
- ** Electronic Industries Association Standard Code.

STANDARD TAPE CHANNEL NUMBERS								Symbol	EIA octal code
8	7	6	5	4	FEED	3	2		1
		X						X	(ZERO) 0 -) 040
									1 - _ 001
							X	X	2 - (z 002
			X				X	X	3 - # 023
						X			4 - \$ 004
			X			X		X	5 - % 025
			X			X		X	6 - ' 026
						X	X	X	7 - & 007
				X					8 - . 010
			X	X				X	9 - (: 031
X	X							X	a - A 141
X	X						X	X	b - B 142
X	X	X					X	X	c - C 163
X	X					X			d - D 144
X	X	X				X		X	e - E 165
X	X	X	X			X	X		f - F 166
X	X	X				X	X	X	g - G 147
X	X			X					h - H 150
X	X		X	X				X	i - I 171
X	X		X				X	X	j - J 121
X	X		X				X	X	k - K 122
X	X	X				X		X	l - L 103
X	X		X			X		X	m - M 124
X	X					X		X	n - N 105
X	X					X	X	X	o - O 106
X	X	X				X	X	X	p - P 127
X	X		X					X	q - Q 130
X				X				X	r - R 111
	X	X					X	X	s - S 062
	X	X					X	X	t - T 043
	X	X	X			X		X	u - U 064
	X	X				X		X	v - V 045
	X	X				X	X	X	w - W 046
	X	X	X			X	X	X	x - X 067
	X	X	X	X					y - Y 070
	X		X	X				X	z - Z 051
									SPACE 020
X									. - " 100
	X	X							/ - ? 061
	X	X	X				X	X	STOP 013
		X	X				X	X	. 133
	X	X	X				X	X	- 073
	X	X	X				X	X	. 153
		X	X			X			PUNCH ON 114
	X	X	X			X			UPPER CASE 174
	X	X	X			X			: - : 160
	X	X	X			X	X		. TAB 076
	X	X	X			X	X		CONTROL 016
	X	X	X			X	X	X	PUNCH OFF 067
		X	X			X	X	X	DATA SELECTOR(AUX. 3) 037
X		X	X			X	X	X	FORM FEED(AUX. L) 117
	X		X			X			
		X	X			X			BACK SPACE 062
		X	X	X		X		X	
	X	X	X			X		X	ADDRESS IDEN(AUX. J) 135
	X	X	X			X	X	X	
	X	X	X			X	X	X	LOWER CASE 172
	X	X	X			X	X	X	
	X	X	X			X	X	X	TAPE FEED 177
X									CAR. RET. 200

100

The following chart shows the ASCII symbols with their octal code equivalents.

Symbol	ASCII Equivalent (octal code)
(Space)	040
!	041
#	043
\$	044
%	045
&	046
'	047
(	050
)	051
*	052
+	053
,	054
-	055
.	056
/	057
0	060
1	061
2	062
3	063
4	064
5	065
6	066
7	067
8	070
9	071
:	072
;	073
<	074
=	075
>	076
?	077
@	100

Symbol	ASCII Equivalent (octal code)
A	101
B	102
C	103
D	104
E	105
F	106
G	107
H	110
I	111
J	112
K	113
L	114
M	115
N	116
O	117
P	120
Q	121
R	122
S	123
T	124
U	125
V	126
W	127
X	130
Y	131
Z	132
[	133
\	134
]	135

The following chart shows the information to be programmed into the conversion table.

Symbol	EIA (octal equiv)	ASCII (octal equiv)	Symbol	EIA (octal equiv)	ASCII (octal equiv)
Sp	020	040	A	141	101
!		041	B	142	102
#		013	C	163	163
\$		044	D	144	104
%		045	E	165	105
&		046	F	166	106
1		047	G	147	107
(		050	H	150	110
)		051	I	171	111
*		052	J	121	112
+		053	K	122	113
,	073	054	L	103	114
-	100	055	M	124	115
.	153	056	N	105	116
/	061	057	O	106	117
0	040	060	P	127	120
1	001	061	Q	130	121
2	002	062	R	111	122
3	023	063	S	62	123
4	004	064	T	43	124
5	025	065	U	64	125
6	026	066	V	45	126
7	007	067	W	46	127
8	010	070	X	67	130
9	031	071	Y	70	131
:		072	Z	57	132
;		073	[		133
<		074	\		134
=		075	]		135
>		076			
?		077			
@		100			
CARR	200	012			

The conversion table portion of a BASIC program is shown on the chart below. Each statement defines one element in the conversion Table A with the use of the OCT function: the octal notation does not need to be translated to decimal. If decimal notation was supplied in the symbol tables, the OCT function could have been omitted; however, it is common to obtain symbol tables in the octal form rather than in decimal form.

```

10 D[M AI[128]
20 A[OCT20]=OCT40
30 A[OCT73]=OCT54
40 A[OCT100]=OCT55
50 A[OCT153]=OCT56
60 A[OCT61]=OCT57
70 A[OCT40]=OCT60
80 A[OCT1]=OCT61
90 A[OCT2]=OCT62
100 A[OCT23]=OCT63
110 A[OCT4]=OCT64
120 A[OCT25]=OCT65
130 A[OCT26]=OCT66
140 A[OCT7]=OCT67
150 A[OCT10]=OCT70
160 A[OCT31]=OCT71
170 A[OCT200]=OCT12
180 A[OCT141]=OCT101
190 A[OCT142]=OCT102
200 A[OCT163]=OCT103
210 A[OCT144]=OCT104
220 A[OCT165]=OCT105
230 A[OCT166]=OCT106
240 A[OCT147]=OCT107
250 A[OCT150]=OCT110
260 A[OCT171]=OCT111
270 A[OCT121]=OCT112
280 A[OCT122]=OCT113
290 A[OCT103]=OCT114
300 A[OCT124]=OCT115
310 A[OCT105]=OCT116
320 A[OCT106]=OCT117
330 A[OCT127]=OCT120
340 A[OCT130]=OCT121
350 A[OCT111]=OCT122
360 A[OCT62]=OCT123
370 A[OCT43]=OCT124
380 A[OCT64]=OCT125
390 A[OCT45]=OCT126
400 A[OCT46]=OCT127
410 A[OCT67]=OCT130
420 A[OCT76]=OCT131
430 A[OCT61]=OCT132

```

ENTER

When a CHAR request is keyed into the calculator for the paper tape reader, conversion is not done and the calculator will display the decimal equivalent of the EIA octal code for the symbol taken from the paper tape. In order to have automatic code conversion, the program must contain an ENTER statement.

410 ENTER (9, 420, A) B

B will be read and its ASCII equivalent found in the conversion table A. If data contained in a string is to be converted to ASCII code, the string name is used instead of the select code, and in this way conversion can be done internally as well as at time of input or output.

OUTPUT

For output of data in a foreign code, automatic code conversion is invoked by use of the OUTPUT statement.

500 OUTPUT (8, 510, A) B

The ASCII B will be found in the conversion table A and changed to the foreign code equivalent before output. Notice that the same conversion table A is used for input and output. The ENTER statement causes the calculator to look for ASCII code, (on the right of the equals (=) signs in the table) and assume that it is

receiving foreign code; whereas the OUTPUT statement causes the calculator to assume it has ASCII code, and to look for the foreign code (on the left of the equals (=) signs in the table above.

ERROR CODES

Program diagnostic or error conditions found in the Extended I/O ROM:

<u>NUMBER</u>	<u>EXPLANATION</u>
ERROR 83	End of data reached or data contains more than ten (10) blanks in a row.
ERROR 84	Invalid format specifications: format must be free format (*), E format or F format.
ERROR 85	Numeric input syntax error: multiple decimal points, more than one E in E format, etc.
ERROR 86	Conversion table not found. Check for integer initialization in DIM statement.

TERMINAL PLUG-IN READ-ONLY MEMORY MODULE

The terminal plug-in read-only memory module that is available with the calculator allows the user to enter, store, and edit free-text. It also allows the user to communicate, either directly or through an external modem, with another calculator, a computer or a time-sharing computer system. The calculator may transmit or receive BASIC language programs or free-text.

To put the calculator into terminal mode, the user types in TERM and actuates the EXECUTE key. If he wants to use the calculator for data transmission, he can specify one or two optional parameters following the TERM mnemonic. The first parameter is the select code of the modem interface module. If another select code is not specified the calculator assumes select code four. The second parameter is the baud rate of the transmitted data. If not specified, the calculator will assume 110 baud which is the same rate as a standard ASR232 teletypewriter. For example, to transmit or receive on select code six at 300 baud, the user enters TERM, 6,300 followed by actuation of the EXECUTE key. The selectable baud rate is continuous in integer increments from 3 to 300. The conversion of characters from parallel to serial format is done automatically within the calculator firmware, so that modem interface circuitry is simplified and no switches are required to change baud rate.

While the calculator is operating in the terminal mode, lines of text may be entered from the keyboard and terminated with the END-OF-LINE key. These lines must be preceded by a line number, but there is no syntax requirement for the remainder of the line. All of the line-by-line and character-by-character editing features of the calculator are available in this mode. These include listing, backspace, forward space, insert character, delete character, and display shift control. Automatic line numbering is also available. In addition, the tape cassette commands, the PTAPE command, the PRINT ALL command, and the LIST command operate normally. The LIST command syntax has been expanded to include LISTX, which means list without line numbers, as well as LIST#SC or LISTX#SC where SC is the select code of the modem interface, which means transmit the information through the modem to the remote system.

When the calculator is operating in the terminal mode, five of the user-definable keys take on special meaning. Key f5 becomes a teletype shift key (TELETYPE is a Registered Trade Mark), and f6 becomes a teletype control key. These keys and the lower case shift key allow the user to generate any seven-bit ASCII code. To generate a teletype shift or control character, the user first actuates the f5 and f6 keys and then actuates the appropriate key on the alpha section of the keyboard. For example, to generate a "control C", the f6 key and the C key are actuated sequentially. No character is entered into the display until after the chosen alpha key is actuated. The character entered into the display may or may not be the same as the alpha character. For instance a "shift 0" generates the symbol—1.

Key f8 is used in the terminal mode to select even or odd parity for transmitted characters. When the terminal mode is first entered, even parity is

assumed. The user may then convert to odd parity by actuating key f8. The display will then indicate ODD. To revert back to even parity the user actuates key f8 again, and the display indicates EVEN.

Key f9 is used in the terminal mode as a transmit key. To transmit a message through the modem interface, the message is typed into the display from the keyboard, and then the line is terminated by actuating key f9. The calculator then serializes the characters entered, and transmits them at the selected baud rate. For example, to obtain a listing of a program from a remote time-sharing computer service, the user enters LIST f9. The transmit key may also be used to enter the responses in a sign-on procedure for a time-sharing service.

Key f7 is used to place the calculator into a mode for saving, in memory, an incoming program. For example, to receive and store a program from a time-sharing computer service, the user actuates key f7 followed by LIST f9. As the program is listed from the time-sharing service, the lines are stored in the calculator's user memory as free-text. If the f7 key is not actuated just prior to entering LIST f9, the program will be printed on the external line printer.

Once a program has been entered as free-text, either from a remote source or from the keyboard, and if it is a BASIC program, it may be checked for syntax errors and converted to BASIC program format in memory. This is done by typing COMP followed by actuation of the EXECUTE key. Any syntax errors will be listed, and only the lines which have correct BASIC syntax will be translated. Incorrect lines will remain in free-text format. After a program has been translated by the COMP command, it may be executed locally using any of the normal execution commands, i.e. RUN, CONT, etc. An attempt to execute an untranslated line will cause an error message (ERROR 79) to be displayed.

The receiving section of the modem driver routine operates under interrupt control which allows the user to execute programs locally and remotely at the same time. For example, the user may want to run a program on a time-sharing computer system that may take several minutes to complete. He may start the program by transmitting a RUN command. While that program is being executed, the calculator is free for normal keyboard operation or program execution. The only limitation in the mode is that this calculator may not use the display or printer at the same time as it is receiving information from the remote program.

PLOTTER PLUG-IN READ-ONLY MEMORY MODULE

The plotter plug-in read-only memory module enables the calculator to control an Hewlett-Packard 9862A calculator plotter, providing permanent graphic solutions to problems solved by the calculator.

In general, the plotter command set can be considered as consisting of two groups: plotting commands and writing commands. The user can specify any plotting units he pleases, the calculator then automatically scales those units to fit the chosen plotting area. Also, the calculator keyboard characters can be drawn in different sizes and directions.

The 'plotting' commands enable the system to automatically scale user-units; draw X and Y axes of any length, anywhere in the plotting area; make any desired tic-marks on the axes; plot points or functions; lower or raise the pen, either before or after movement; temporarily translate the established origin to any point within the plotting area and then plot, still in user-units, with respect to the new origin; plot in increments (that is, in user-units, plot any point with respect to the current pen position).

The 'writing' commands enable most calculator keyboard characters to be 'printed' on the plotter. The user can specify the position, height and width of the characters, and the direction in which they will be printed. A 'centering' command (CPLOT) enables labels to be centered on some particular point, thus simplifying labelling of axes and of specific points on the graph. The format of labels and numbers — field width, fixed or floating point, the number of digits following the decimal point, etc. — is specified by standard FORMAT statements. In addition, a unique LETTER command establishes a 'typewriter' mode enabling the plotter to be controlled and positioned from the calculator keyboard, on a character-by-character basis; this allows the user to add extra labelling or individual comments to his graphs.

INITIALIZING THE PLOTTER

Before plotting, the plotter must be prepared and the physical limits of the

plotting area must be established. The front-panel controls on the plotter are used for this purpose.

LINE AND CHART HOLD

The LINE pushbutton is the power switch for the plotter; press it to apply power, and press it again to remove power; the white LINE lamp lights whenever the plotter is ON.

Pressing CHART HOLD activates the electro-static paper hold-down mechanism. Pressing CHART HOLD again deactivates it. The plotter will not plot or letter, and the pen holder and arm will move freely in all directions when CHART HOLD is deactivated.

LOADING PAPER

To load paper, release CHART HOLD and manually move the pen arm all the way to one side of the plotter. Lay a sheet of paper on the plotting surface and smooth out any irregularities in the paper (you may also wish to ensure that the paper is squarely against the ridge at the bottom of the plotting surface); then activate CHART HOLD.

GRAPH LIMITS

The graph limit controls are used to determine the physical size of the plot. LOWER LEFT and the two knobs to its left are used to determine the physical location of the lower left hand corner of the plotting area.

UPPER RIGHT and the two knobs to its right are used to determine the physical location of the upper right-hand corner of the plotting area. Together, the upper right-hand corner and the lower left-hand corner determine the size of the plotting area.

Also, altering the lower left-hand setting will translate the upper right-hand setting by the same direction and amount.

To specify the lower-left hand corner of the plotting area, press LOWER LEFT; the pen will move (without touching the paper) to the lower left-hand corner of the plotting area. This point can be set anywhere within the lower left-hand quarter of the plotting surface (platen) by adjusting the two knobs associated with LOWER LEFT. Once the lower left-hand corner has been set, the upper right-hand corner is set in the same general way by pressing UPPER RIGHT and adjusting the two knobs associated with it. Once the plotting area has been determined, it can be relocated by moving the position of the lower left-hand corner — the upper right-hand corner will 'track' the change.

PLOTTING COMMANDS

- NOTES: 1. All commands can be activated either from the keyboard or from a program except where noted.
2. All values in the following statements can be numbers, variables or expressions except where noted.
3. Any parameter enclosed in square brackets is optional as far as the statement containing it is concerned. However, program sense may dictate that the parameter be present in specific cases.

THE SCALE STATEMENT

SCALE Xmin, Xmax, Ymin, Ymax
Examples:

SCALE -10, 10, -5, 5

SCALE 4PI, 4PI, 3, 1.1

establishes the full-scale units for the plot. Xmin to Xmax and Ymin to Ymax correspond exactly to the limits of the horizontal and vertical edges, respectively, of the plotting area (the area is established mechanically as previously described). This also establishes the point, on or off the plotting area, where the origin of the graph (0, 0) is located.

A SCALE statement must be executed before any plotting can occur. Once established the scale remains established until one of the following occurs:

A new SCALE statement is executed.

The program is initialized.

A SCRATCH or SCRATCH A or SCRATCH V is executed.

The calculator is switched off.

The parameters (X, Y, etc.) in the SCALE statement must be given in the correct order. If the minimum or maximum values are switched no ERROR message will occur; however, subsequent plotting commands may not be executed properly.

The SCALE statement has no effect on the position of the pen.

THE PEN STATEMENT PEN

The PEN statement is a 'stand-alone' instruction requiring no parameter. It raises the pen without otherwise changing its position relative to the plotting area.

Instructions to raise or lower the pen, either before or after movement, can be easily included in several other statements (see PLOT and IPLOT so there is no special 'lower pen' instruction).

THE OFFSET STATEMENT OFFSET X, Y

Example: OFFSET 3, -3

Temporarily offsets the origin (point 0, 0 established by the previous SCALE statement) by an amount, and in the direction, determined by the values (in user-units) of X and Y. All future plotting commands are then made with respect to the new origin until such time as that origin is again changed by means of, for example, a new OFFSET or a new SCALE statement.

The OFFSET statements are not accumulative; that is, a new "offset" is with respect to the original origin and not with respect to the last offset origin.

Offsetting greatly simplifies plotting, from the user's point of view, when it becomes necessary to divide the plotting area into several smaller segments and then make a separate plot in each segment. As the plot is made in each segment it is not necessary for the user to 'correct' each point before plotting; instead OFFSET statement moves the origin to some convenient point within that segment so that the calculator automatically makes the necessary 'corrections' for each point plotted.

THE AXIS STATEMENT

X AXIS Y-offset [, $\pm$ tic [, start point, end point]]
or Y AXIS X-offset [, $\pm$ tic [, start point, end point]]
X AXIS 3, 1, -4, 4

Draws an X-(or Y-) axis according to the parameters given in the AXIS statement. The pen is automatically raised both before and after drawing the axis. (NOTE: The following describes the X-axis; the same information is applicable to the Y-axis if 'left' and 'right' for the X-axis are read as, respectively, 'bottom' and 'top' for the Y-axis.)

1. If no optional parameters are given, draws a straight line from left to right across the complete plotting area (from Xmin to Xmax). The line crosses the Y-axis at a point determined by the value of "y-offset".
2. If a 'tic' parameter is included then tic marks are made along the axis as it is drawn; the value for the 'tic' determines the spacing, in user-units, between tics. The first tic is drawn at the starting point of the line. The tic parameter is usually positive (the sign is not required), but occasionally a negative tic spacing is useful — see 4, below.
3. If the start point/end point parameters are given, then the axis is drawn only between the points specified — from the start point to the end point.
4. a. A negative tic spacing when no start point/end point parameters are given results in a tic only at the left end (Xmin) of the axis.
b. If the start point parameter is more positive than (i.e., to the right of) the end point parameter, then the axis is drawn from right to left; in this case, negative tic spacing results in normal tic marks being drawn along the axis.
c. With the start point/end point parameters the same as in b above, a positive tic spacing results in a tic only at the right end (Xmax) of the axis.

THE PLOT STATEMENT

PLOT X, Y [, control pen]
PLOT SIN(X), COS(X), -2

Moves the pen to the co-ordinate specified by the value of X and Y.
When no optional 'control pen' parameter is given:

If the pen was raised, it moves to the point specified and then lowers.
If the pen was lowered, it remains lowered while moving to the point specified, thus drawing a line on the plotting surface.

The 'Control Pen' Parameter

5 The value and sign of this parameter in the PLOT (and IPLOT) statements determines whether the pen will be raised or lowered before or after it moves to the specified point.
If the parameter is:

- 10 negative — control occurs after movement;
- positive — control occurs before movement;
- odd — raises pen if it was lowered;
- even — lowers pen if it was raised.

15 The value of the control parameter can be any number in the range ± 32767 . If the value is not an integer then it is automatically rounded up or down according to the value of the fractional part of the number; that is up to .5 or greater, or down for less than .5. (Rounding is the same as the standard rounding in the calculator; it is not the same as the INT function, where the value becomes that of the next lower integer.)

THE IPLOT STATEMENT

20 IPLOT deltaX, deltaY [, control pen]
IPLOT 2, -3A/4, 1

Moves the pen (from its current position) in the X direction and in the Y direction, by the amounts specified by 'deltaX' and 'deltaY', respectively.

25 The 'control pen' parameter is optional and operates exactly as described previously — see the PLOT STATEMENT.

Notice that the action of the IPLOT statement is such that it is as if, during the execution of that statement only, the origin (0, 0) of the graph is offset to the current position of the pen. Pen movement is then related to that offset origin.

30 The IPLOT statement is most useful when drawing regular geometric shapes such as, for example, a swastika (卐). In this case each point is more easily plotted with respect to the previous point, rather than with respect to the origin of the graph.

THE LABEL STATEMENT

35 LABEL (FORMAT statement number or *[, character height in %, Aspect ratio, angle of rotation [, paper height/paper width]]) print list
Examples: LABEL (x, 2, 2, 0, 8.5/11) "PLOTTER"
LABEL (100) 1, A, SIN(X)

40 The LABEL statement is used to write alpha and numeric characters with the plotter. Several parameters are allowed which can be used to control the size, shape, and angle of rotation of the character printed. The character height can be specified in percent of the paper height. The aspect ratio is the ratio of the character height to the character width before any rotation. The angle of rotation of characters printed can be given. This parameter can be given in degrees, radians, or grads and is dependent on a previously given DEG, RAD, or GRAD statement. A fourth parameter can be specified. This is the ratio of the actual measured height of the plot paper to the measured width as set by the upper right and lower left positions. This parameter is necessary to keep proper aspect ratio of characters printed on an angle on a non-square plot. If not specified, the calculator will assume character height = 2.5%, aspect ratio = 2, rotation = 0, and paper ratio = 1.

50 The format specification and the print list are the same as a WRITE statement. If a FORMAT statement number is specified, the print list is written on the plotter using the specification given in the FORMAT statement. All specifications are allowed except "B". If an "*" is used, the print list will be written according to standard format. This includes the normal definition of comma and semi-colon spacing, string fields, TAB, etc.

55 The LABEL statement will start printing characters at the current pen position. Anytime an end of line is needed, the plotter pen will return to the character position directly below the first character of the current line, simulating a carriage return, line feed.
60

100

If the string variables option block is also plugged into the memory, then string variables are allowed in the print list.

THE LETTER STATEMENT

LETTER

When the LETTER statement is executed, the calculator enters a unique 'typewriter' mode with the plotter as the printing device. While in this mode when the user hits any printing character on the keyboard, that character is immediately printed on the plotter at the current pen position. An EOL or EXECUTE will cause a carriage return, line-feed to be simulated. In addition, while in this mode the $\uparrow$, $\downarrow$, $\leftarrow$, and $\rightarrow$ keys can be used to position the pen. The $\uparrow$ and $\downarrow$ keys can cause the pen to move up or down one character position. The $\leftarrow$ and $\rightarrow$ keys cause the pen to move left or right one character position. If the shift key is held down at the same time as a pen control key, the pen will move one-tenth of a character position. Character size, aspect ratio, and rotation can be specified by giving a LABEL statement prior to the LETTER statement.

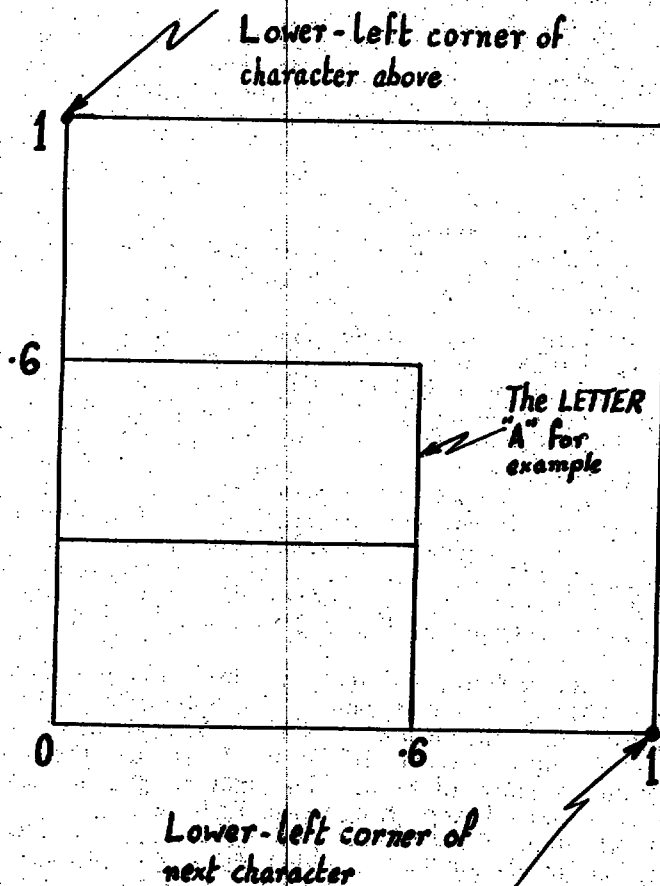
THE CPLOT STATEMENT

CPLOT delta X characters, delta Y characters

Example: CPLOT 5, -3

The CPLOT statement is similar to the IPLOT statement in that it moves the pen to a position relative to the current pen position. The difference is that the delta X and delta Y values are specified in character size units. In the example above the pen would move five character position down. This statement is particularly useful in positioning the pen when labeling a plot using the LABEL statement.

One character space is defined as follows:





- The error messages given by the plotter module are:
- ERROR 80 — No scale statement executed before PLOT, IPLOT, OFFSET, or AXIS.
- ERROR 81 — Character size too large (limited to = 20%) or binary mode not allowed.
- ERROR 82 — OFFSET, POINT 1, or POINT 2 out of range during axis execution or tic increment in axis statement is too small.

MATRIX PLUG-IN READ-ONLY MEMORY MODULE

The Matrix read-only memory module enables the calculator to understand the MAT statements of BASIC. These statements facilitate the matrix operations of addition, subtraction, initialization, scalar multiplication, matrix multiplication transposition, and inversion. A function additional to standard BASIC language matrix operations is provided for computing the determinant of a square matrix. The MAT READ and MAT PRINT commands facilitate entering data into an array and the printing thereof. The MAT INPUT command which appears in some versions of BASIC is not allowed. The matrix operations are allowed on split precision or integer arrays as well as full floating point arrays.

The operations performed by this matrix module are summarized below and additional information is provided above in section IV. above.

A. MAT READ

reads numeric information from DATA statements into an array

MAT READ A

MAT READ A(3,5)

MAT READ B,W(8),X,Y

B. MAT PRINT

prints complete arrays

close packing specified with a semicolon

MAT PRINT A

MAT PRINT A;

MAT PRINT R,S;V;W

C. ZER, CON, and IDN operations

ZER: initialize an array to all zeros

CON: initialize an array to all ones

IDN: initialize an array to the identity matrix

MAT A = ZER

MAT Q = CON(9)

MAT I = IDN(4,4)

D. MAT assignment statements

assign to a matrix the result of a MAT operation

1) MAT A = B

assigns elements of A from array B

124

100

1,444,141

100

2) MAT A = B ± C

performs indicated addition or subtraction and assigns
result to A

any of A, B, and C may be the same matrix

5

3) MAT A = B * C

5

performs indicated multiplication and assigns result to A

array A must be distinct from B or C

4) MAT A = (expression) * B

10

performs the indicated scalar multiply (each element of B is multiplied by
the value of the expression) and assigns the result to A

10

5) MAT A = TRN(B)

assigns the transpose of B to A

array A must be distinct from B

6) MAT A = INV(B)

15

assigns the inverse of B to A

15

A and B may be the same matrix

E. DET operation

calculates the determinant of a square matrix

MAT D4 = DET(A)

20

KEY CODES AND MNEMONICS

20

25

All of the keys of the keyboard input unit and their associated mnemonics and binary keycodes are listed in the Table below. Every key has one mnemonic and two keycodes, (namely, a shifted keycode and an unshifted keycode). Keycodes are applied to the CPU in eight-bit binary form. The first four bits of each keycode are given in the left-most vertical column of the table below and the next three bits of each keycode are given in the uppermost row of the table below. The eighth bit of each keycode is the shift bit and is determined by whether or not the shift key of the keyboard input unit is depressed. Keycodes entered into the CPU from the keyboard input unit or from the program storage section of the memory unit are processed by the keyboard input routine 204 as generally described above in connection with Figure 9 and as shown in detail in Figures 88A—G.

25

30

30

				D0	D1	D2	D3	D4	D5	D6	D7
b6	0	0	0	0	0	0	1	1	1	1	1
b5	0	0	0	1	1	1	0	0	0	1	1
b4	0	1	0	0	1	1	0	1	1	0	1
b3b2b1b0											
0 0 0 0	f0	RECALL	SPACE	Ø			P		PA		
0 0 0 1	f1	FETCH		1	A	Q		STOP			
0 0 1 0	f2	BACK		2	B	R		EOL			
0 0 1 1	f3	FWD		3	C	S		DL			
0 1 0 0	f4	↓		4	D	T		FXD			
0 1 0 1	f5	↑		5	E	U		FLT			
0 1 1 0	f6	←		6	F	V		SCRATCH			
0 1 1 1	f7	→		7	G	W		AUTO			
1 0 0 0	f8	LOAD	(	8	H	X					
1 0 0 1	f9	STORE	)	9	I	Y					
1 0 1 0	LIST	INIT	X	*	J	Z		CLR			
1 0 1 1	EXEC	/	+	†	K			RESULT			
1 1 0 0	CONT		,		L						
1 1 0 1	STEP	STD	-	=	M						
1 1 1 0	TRACE	NORMAL	.		N	↑					
1 1 1 1	RUN	INSERT	? /		O			ENTER EXP			

BASIC INSTRUCTION SET

Every routine and subroutine of the calculator comprises a sequence of one or more of 71 basic sixteen-bit instructions listed below. These 71 instructions are all implemented serially by the micro-processor in a time period which varies according to the specific instruction, to whether or not it is indirect, and to whether or not the skip condition has been met.

Upon completion of the execution of each instruction, the program counter (P register) has been incremented by one except for instructions JMP, JSM; and the skip instructions in which the skip condition has been met. The M-register is left with contents identical to the P-register. The contents of the addressed memory location and the A and B registers are left unchanged unless specified otherwise.

Memory Reference Group

Memory Reference Group
The 14 memory reference instructions refer to a specific address in memory determined by the address field <m>, by the ZERO/CURRENT page bit, and by the DIRECT/INDIRECT bit. Page addressing and indirect addressing are both described in detail in the reference manuals for the Hewlett-Packard Model 2116 computer (hereinafter referred to as the HP 2116).

The address field <m> is a 10 bit field consisting of bits 0 through 9. The ZERO/CURRENT page bit is bit 10 and the DIRECT/INDIRECT bit is bit 15, except for reference to the A or B register in which case bit 8 becomes the DIRECT/INDIRECT bit. An indirect reference is denoted by a <, I> following the address <m>.

Memory Reference Group (continued).

REGISTER REFERENCE OF A OR B REGISTER: If the location <A> or is used in place of <m> for any memory reference instruction, the instruction will treat the contents of A or B exactly as it would the contents of location <m>. See the note below on the special restriction for direct register reference of A or B.

ADA m, I Add to A. The contents of the addressed memory location m are added (binary add) to contents of the A register, and the sum remains in the A register. If carry occurs from bit 15, the E register is loaded with 0001, otherwise E is left unchanged.

ADB m, I Add to B. Otherwise identical to ADA.

CPA m, I Compare to A and skip if unequal. The contents of the addressed memory location are compared with the contents of the A register. If the two 16-bit words are different, the next instruction is skipped; that is, the P and M registers are advanced by two instead of one. Otherwise, the next instruction will be executed in normal sequence.

CPB m, I Compare to B and skip if unequal. Otherwise identical to CPA.

LDA m, I Load into A. The A register is loaded with the contents of the addressed memory location.

LDB m, I Load into B. The B register is loaded with the contents of the addressed memory location.

STA m, I Store A. The contents of the A register are stored into the addressed memory location. The previous contents of the addressed memory location are lost.

STB m, I Store B. Otherwise identical to STA.

IOR m, I "Inclusive OR" to A. The contents of the addressed location are combined with the contents of the A register as an "INCLUSIVE OR" logic operation.

ISZ m, I Increment and Skip if Zero. The ISZ instruction adds ONE to the contents of the addressed memory location. If the result of this operation is ZERO, the next instruction is skipped; that is, the P and M registers are advanced by TWO instead of ONE. The incremental value is written back into the addressed memory location. Use of ISZ with the A or B register is limited to indirect reference; see footnote on restrictions.

AND m, I Logical "AND" to A. The contents of the addressed location are combined with the contents of the A register as an "AND" logic operation.

DSZ m, I Decrement and Skip if Zero. The DSZ instruction subtracts ONE from the contents of the addressed memory location. If the result of this operation is zero, the next instruction is skipped. The decremented value is written back into the addressed memory location. Use of DSZ with the A or B register is limited to indirect reference; see footnote on restrictions.

JSM m, I Jump to Subroutine. The JSM instruction permits jumping to a subroutine in either ROM or R/W memory. The contents of the P register is stored at the address contained in location 1777 (stack pointer). The contents of the stack pointer is incremented by one, and both M and P are loaded with the referenced memory location.

JMP m, I Jump. This instruction transfers control to the contents of the addressed location. That is, the referenced memory location is loaded into both M and P registers, effecting a jump to that location.

Shift-Rotate Group

The eight shift-rotate instructions all contain a 4 bit variable shift field $\langle n \rangle$ which permits a shift of one through 16 bits; that is, $1 \leq n \leq 16$. If $\langle n \rangle$ is omitted, the shift will be treated as a one bit shift. The shift code appearing in bits 8, 7, 6, 5 is the binary code for $n-1$, except for SAL and SBL, in which cases the complementary code for $n-1$ is used.

- 5 AAR n Arithmetic right shift of A. The A register is shifted right n places with the sign bit (bit 15) filling all vacated bit positions. That is, the $n+1$ most significant bits become equal to the sign bit. 5
- 10 ABR n Arithmetic right shift of B. Otherwise identical to AAR. 10
- SAR n Shift A right. The A register is shifted right n places with all vacated bit positions cleared. That is, the n most significant bits become equal to zero.
- SBR n Shift B right. Otherwise identical to SAR.
- 15 SAL n Shift A left. The A register is shifted left n places with the n least significant bits equal to zero. 15
- SBL n Shift B left. Otherwise identical to SAL.
- RAR n Rotate A right. The A register is rotated right n places, with bit 0 rotated around to bit 15.
- 20 RBR n Rotate B right. Otherwise identical to RAR. 20

Alter-Skip Group

The sixteen alter-skip instructions all contain a 5-bit variable skip field $\langle n \rangle$ which, upon meeting the skip condition, permits a relative branch to any one of 32 locations. Bits 9, 8, 7, 6, 5 are coded for positive or negative relative branching in which the number $\langle n \rangle$ is the number to be added to the current address, (skip in forward direction), and the number $\langle -n \rangle$ is the number to be subtracted from the current address, (skip in negative direction). If $\langle n \rangle$ is omitted, it will be interpreted as a ONE.

	$\langle n \rangle = 0$	CODE=00000	REPEAT SAME INSTRUCTION	
	$\langle n \rangle = 1$	CODE=00001	DO NEXT INSTRUCTION	
	$\langle n \rangle = 2$	CODE=00010	SKIP ONE INSTRUCTION	
	$\langle n \rangle = 15$	CODE=01111	ADD 15 TO ADDRESS	
	$\langle n \rangle = -1$	CODE=11111	DO PREVIOUS INSTRUCTION	
	$\langle n \rangle = -16$	CODE=10000	SUBTRACT 16 FROM ADDRESS	
	$\langle n \rangle = \text{nothing}$	CODE=00001	DO NEXT INSTRUCTION	

The alter bits consist of bits 10 and bits 4. The letter $\langle S \rangle$ following the instruction places a ONE in bit 10 which causes the tested bit to be *set* after the test. Similarly the letter $\langle C \rangle$ will place a ONE in bit 4 to *clear* the test bit. If both a set and clear bit are given, the set will take precedence. Alter bits do not apply to SZA, SZB, SIA, and SIB.

- SZA n Skip if A zero. If all 16 bits of the A register are zero, skip to location defined by n .
- SZB n Skip if B zero. Otherwise identical to SZA.
- 45 RZA n Skip if A not zero. This is a "Reverse Sense" skip of SZA. 45
- RZB n Skip if B not zero. Otherwise identical to RZA.
- SIA n Skip if A zero; then increment A. The A register is tested for zero, then incremented by one. If all 16 bits of A were zero before incrementing, skip to location defined by n .
- 50 SIB n Skip if B zero; then increment B. Otherwise identical to SIA. 50

(16)

Alter-Skip Group (continued)

RIA n Skip if A not zero; then increment A. This is a "Reverse Sense" skip of SIA.

RIB n Skip if B not zero; then increment B. Otherwise identical to RIA.

5 SLA n, S/C Skip if Least Significant bit of A is zero. If the least significant bit (bit 0) of the A register is zero, skip to location defined by n. If either S or C is present, the test bit is altered accordingly after test.

SLB n, S/C Skip if Least Significant bit of B is zero. Otherwise identical to SLA.

10 SAM n, S/C Skip if A is Minus. If the sign bit (bit 15) of the A register is a ONE, skip to location defined by n. If either S or C is present, bit 15 is altered after the test.

SBM n, S/C Skip if B is Minus. Otherwise identical to SAM.

15 SAP n, S/C Skip if A is Positive. If the sign bit (bit 15) of the A register is a ZERO, skip to location defined by n. If either S or C is present, bit 15 is altered after the test.

SBP n, S/C Skip if B is Positive. Otherwise identical to SAP.

20 SES n, S/C Skip if Least Significant bit of E is Set. If bit 0 of the E register is a ONE, skip to location defined by n. If either S or C is present, the entire E register is set or cleared respectively.

SEC n, S/C Skip if Least Significant bit of E is Clear. If bit 0 of the E register is a ZERO, skip to location defined by n. If either S or C is present, the entire E register is set or cleared respectively.

Complement-Execute-DMA Group

25 These seven instructions include complement operations and several special-purpose instructions chosen to speed up printing and extended memory operations.

CMA Complement A. The A register is replaced by its One's complement.

CMB Complement B. The B register is replaced by its One's complement.

30 TCA Two's Complement A. The A register is replaced by its One's Complement and incremented by one.

TCB Two's complement B. The B register is replaced by its One's Complement and incremented by one.

35 EXA Execute A. The contents of the A register are treated as the current instruction, and executed in the normal manner. The A register is left unchanged unless the instruction code causes A to be altered.

EXB Execute B. Otherwise identical to EXA.

40 DMA Direct Memory Access. The DMA control in Extended Memory is enabled by setting the indirect bit in M and giving a WTM instruction. The next ROM clock transfers A→M and the following two cycles transfer B→M. ROM clock then remains inhibited until released by DMA control.

Note: Special Restriction for Direct Register Reference of A or B.

45 For the five register reference instructions which involve a write operation during execution, a register reference to A or B must be restricted to an INDIRECT reference. These instructions are STA, STB, ISZ, DSZ, and JSM. A DIRECT register reference to A or B with these instructions may result in

program modification. (This is different from the hp 2116 in which a memory reference to the A or B register is treated as a reference to locations 0 or 1 respectively.) A reference to location 0 or 1 will actually refer to locations 0 or 1 in Read Only Memory.

5 Input/Output Group (IOG) 5

The eleven IOG instructions, when given with a select code, are used for the purpose of checking flags, setting or clearing flag and control flip-flops, and transferring data between the A/B registers and the I/O register.

10 STF <SC> Set the flag. Set the flag flip-flop of the channel indicated by select code <SC>. 10

CLF <SC> Clear the flag flip-flop of the channel indicated by select code <SC>.

SFC <SC> Skip if flag clear. If the flag flip-flop is clear in the channel indicated by <SC>, skip the next instruction.

15 SFS <SC> H/C Skip if flag set. If the flag flip-flop is set in the channel indicated by <SC>, skip the next instruction. H/C indicates if the flag flip-flop should be held or cleared after executing SFS. 15

CLC <SC> H/C Clear control. Clear the control flip-flop in the channel indicated by <SC>. H/C indicates if the flag flip-flop should be held or cleared after executing CLC. 20

STC <SC> H/C Set Control. Set the control flip-flop in the channel indicated by <SC>. H/C indicates if the flag flip-flop should be held or cleared after executing STC.

25 OT* <SC> H/C Output A or B. Sixteen bits from the A/B register are output to the I/O register. H/C allows holding or clearing the flag flip after execution of OT*. The different select codes allow different functions to take place after loading the I/O register. 25

SC=00 Data from the A or B register is output eight bits at a time for each OT* instruction given. The A or B register is rotated right eight bits. 30

SC=01 The I/O register is loaded with 16 bits from the A/B registers.

SC=02 Data from the A/B register is output one bit at a time for each OT* instruction for the purpose of giving data to the Magnetic Card Reader. The I/O register is unchanged. 35

SC=04 The I/O register is loaded with 16 bits from the A/B register and the control flip flop for the printer is then set.

SC=08 The I/O register is loaded with 16 bits from the A/B register and the control flip flop for the display is then set. 40

SC=16 The I/O register is loaded with 16 bits from the A/B register and then data in the I/O register is transferred to the switch latches. 45

LI* <01> H/C Load into A or B. Load 16 bits of data into the A/B register from the I/O register. H/C allows holding or clearing the flag flop after LI* has been executed.

Input-Output Group (IOG), continued.

LI* <00> The least significant 8 bits of the I/O register are loaded into the most significant locations in the A or B register.

MI* <01> H/C Merge into A or B. Merge 16 bits of data into the A/B register from the I/O register by "inclusive or". H/C allows holding or clearing the flag flop after MI* has been executed.

MI* <00> The least significant 8 bits of the I/O register are combined by inclusive OR with the least significant 8 bits of the A or B register, and rotated to the most significant bit locations of the A or B register.

MAC Instruction Group

A total of 16 MAC instructions are available for operation:

- (a) with the whole floating-point data (like transfer, shifts, etc), or
- (b) with two floating-point data words to speed up digit and word loops in arithmetic routines.

NOTE: <A₀₋₃> means: contents of A-register bit 0 to 3

AR 1 is a mnemonic for arithmetic pseudo-register located in R/W memory on addresses 1744 to 1747 (octal)

AR 2 is a mnemonic for arithmetic pseudo-register located in R/W memory on addresses 1754 to 1757 (octal)

D_i means: mantissa i-th decimal digit; most significant digit is D₁ least significant digit is D₁₂ decimal point is located between D₁ and D₂

Every operation with mantissa means BCD-coded decimal operation.

RET Return
16-bit-number stored at highest occupied address in stack is transferred to P- and M-registers. Stack pointer (=next free address in stack) is decremented by one. <A>, , <E> unchanged.

MOV Move overflow
The contents of E-register is transferred to A₀₋₃. Rest of A-register and E-register are filled by zeros. unchanged.

CLR Clear a floating-point data register in R/W memory on location <A>
ZERO → <A>, <A>+1, <A>+2, <A>+3
<A>, , <E> unchanged

XFR Floating-point data transfer in R/W memory from location <A> to location .
Routine starts with exponent word transfer.
Data on location <A> is unchanged.
<E> unchanged.

MRX AR1 mantissa is shifted to right n-times. Exponent word remains unchanged.

<B₀₋₃> = n (binary coded)

1st shift: <A₀₋₃> → D₁; D₁ → D₁₊₁; D₁₂ is lost

jth shift: 0 → D₁; D₁ → D₁₊₁; D₁₂ is lost

nth shift: 0 → D₁; D₁ → D₁₊₁; D₁₂ → A₀₋₃

0 → E, A₄₋₁₅

each shift: <B₀₋₃> - 1 → B₀₋₃

<B₄₋₁₅> unchanged

MRY AR2 mantissa is shifted to right n-times. Otherwise identical to MRX

MLS AR2 mantissa is shifted to left once. Exponent word remains unchanged.

0 → D₁₂; D₁ → D₁₋₁; D₁ → A₀₋₃

 unchanged

	DRS	AR1 mantissa is shifted to right once. Exponent word remains unchanged $0 \rightarrow D_1; D_1 \rightarrow D_{1+1}; D_{12} \rightarrow A_{0-3}$ ZERO $\rightarrow$ E and A_{4-15} unchanged	
5	DLS	AR1 mantissa is shifted to left once. Exponent word remains unchanged. $\langle A_{0-3} \rangle \rightarrow D_{12}; D_1 \rightarrow D_{1-1}; D_1 \rightarrow A_{0-3}$ $0 \rightarrow E, A_{4-15}$ unchanged	5
10	FXA	Fixed-point addition Mantissas in pseudo-registers AR2 and AR1 are added together and result is placed into AR2. Both exponent words remain unchanged. When overflow occurs "0001" is set into E-reg., in opposite case <E> will be zero.	10
15		$\langle AR2 \rangle + \langle AR1 \rangle + DC \rightarrow AR2$ DC = 0 if <E> was 0000 before routine execution DC = 1 if <E> was 1111 before routine execution , <AR1> unchanged	15
20	FMP	Fast multiply Mantissas in pseudo-registers AR2 and AR1 are added together <B ₀₋₃ >-times and result is placed into AR2. Total decimal overflow is placed to A ₀₋₃ . Both exponent words remain unchanged.	20
25		$\langle AR2 \rangle + \langle AR1 \rangle * \langle B_{0-3} \rangle + DC \rightarrow AR2$ DC = 0 if <E> was 0000 before routine execution DC = 1 if <E> was 1111 before routine execution ZERO $\rightarrow$ E, A_{4-15} <AR1> unchanged	25
30	FDV	Fast divide Mantissas in pseudo-registers AR2 and AR1 are added together so many times until first decimal overflow occurs. Result is placed into AR2. Both exponent words remain unchanged. Each addition without overflow causes +1 increment of .	30
35		1st addition: $\langle AR2 \rangle + \langle AR1 \rangle + DC \rightarrow AR2$ DC = 0 if <E> was 0000 before routine execution DC = 1 if <E> was 1111 before routine execution next additions: $\langle AR2 \rangle + \langle AR1 \rangle \rightarrow AR2$ ZERO $\rightarrow$ E <AR1> unchanged	35
40	CMX	10's complement of AR1 mantissa is placed back to AR1, and ZERO is set into E-register. Exponent word remains unchanged unchanged	40
	CMY	10's complement of AR2 mantissa. Otherwise identical to CMY	
45	MDI	Mantissa decimal increment. Mantissa on location <A> is incremented by decimal ONE on D ₁₂ level, result is placed back into the same location, and zero is set into E-reg. Exponent word is unchanged.	45
50		When overflow occurs, result mantissa will be 1, 000 0000 0000 (dec) and 0001 (bin) will be set into E-reg. unchanged.	50

NRM Normalization

Mantissa in pseudo-register AR2 is rotated to the left to get $D_1 \neq 0$. Number of these 4-bit left shifts is stored in B_{0-3} in binary form ($\langle B_{4-15} \rangle = 0$)

when $\langle B_{0-3} \rangle = 0, 1, 2, \dots, 11$ (dec) $\rightarrow \langle E \rangle = 0000$

When $\langle B_{0-3} \rangle = 12$ (dec) $\rightarrow$ mantissa is zero, and $\langle E \rangle 0001$

Exponent word remains unchanged

$\langle A \rangle$ unchanged.

The binary codes of all of the above instructions are listed in the following coding table, where * implies the A or B register, D/I means direct/indirect, A/B means A register/B register, Z/C means zero page (base page) (current page), H/S means hold test bit/set test bit, and H/C means hold test bit/clear test bit. D/I, A/B, Z/C, H/S, and H/C are all coded as 0/1.

CODING TABLE

GROUP	OCTAL	INSTR	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEMORY REFERENCE GROUP	-0----	AD*	$\frac{D}{I}$	0	0	0	$\frac{A}{B}$	$\frac{Z}{C}$	← MEMORY ADDRESS →									
	-1----	CP*	$\frac{D}{I}$	0	0	1	$\frac{A}{B}$	$\frac{Z}{C}$										
	-2----	LD*	$\frac{D}{I}$	0	1	0	$\frac{A}{B}$	$\frac{Z}{C}$										
	-3----	ST*	$\frac{D}{I}$	0	1	1	$\frac{A}{B}$	$\frac{Z}{C}$										
	-4----	IOR	$\frac{D}{I}$	1	0	0	0	$\frac{Z}{C}$										
	-4----	ISZ	$\frac{D}{I}$	1	0	0	1	$\frac{Z}{C}$										
	-5----	AND	$\frac{D}{I}$	1	0	1	0	$\frac{Z}{C}$										
	-5----	DSZ	$\frac{D}{I}$	1	0	1	1	$\frac{Z}{C}$										
SHIFT- ROTATE GROUP	07---0	A*R	0	1	1	1	$\frac{A}{B}$	← SHIFT CODE →		0 0 0 0								
	07---2	S*R	0	1	1	1	$\frac{A}{B}$			0 0 1 0								
	07---4	S*L	0	1	1	1	$\frac{A}{B}$			0 1 0 0								
	07---6	R*R	0	1	1	1	$\frac{A}{B}$			0 1 1 0								
ALTER- SKIP GROUP	07---0	SZ*	0	1	1	1	$\frac{A}{B}$	0	← SKIP CODE →		0 1 0 0 0							
	07---0	RZ*	0	1	1	1	$\frac{A}{B}$	1			0 1 0 0 0							
	07---0	SI*	0	1	1	1	$\frac{A}{B}$	0			1 1 0 0 0							
	07---0	RI*	0	1	1	1	$\frac{A}{B}$	1			1 1 0 0 0							
	07---1	SL*	0	1	1	1	$\frac{A}{B}$	$\frac{H}{S}$			$\frac{H}{C}$ 1 0 0 1							
	07---2	S*M	0	1	1	1	$\frac{A}{B}$	$\frac{H}{S}$			$\frac{H}{C}$ 1 0 1 0							
	07---3	S*P	0	1	1	1	$\frac{A}{B}$	$\frac{H}{S}$			$\frac{H}{C}$ 1 0 1 1							
	07---4	SES	0	1	1	1	$\frac{A}{B}$	$\frac{H}{S}$			$\frac{H}{C}$ 1 1 0 0							
REGISTER REFERENCE GROUP	07---17	APA	0	1	1	1	$\frac{A}{B}$	—	$\frac{D}{I}$	0	0	0	0	1	1	1	1	1
	07---37	ADB	0	1	1	1	$\frac{A}{B}$	—	$\frac{D}{I}$	0	0	0	1	0	1	1	1	1
	07---57	CPA	0	1	1	1	$\frac{A}{B}$	—	$\frac{D}{I}$	0	0	0	1	1	1	1	1	1
	07---77	CPB	0	1	1	1	$\frac{A}{B}$	—	$\frac{D}{I}$	0	0	0	1	1	1	1	1	1
	07---17	LDA	0	1	1	1	$\frac{A}{B}$	—	$\frac{D}{I}$	0	1	0	0	0	1	1	1	1
	07---37	LDB	0	1	1	1	$\frac{A}{B}$	—	$\frac{D}{I}$	0	1	0	1	0	1	1	1	1
	07---57	STA	0	1	1	1	$\frac{A}{B}$	—	1	0	1	1	0	1	1	1	1	1
	07---577	STB	0	1	1	1	$\frac{A}{B}$	—	1	0	1	1	1	1	1	1	1	1
	07---17	IOR	0	1	1	1	$\frac{A}{B}$	—	$\frac{D}{I}$	1	0	0	0	0	1	1	1	1
	07---637	ISZ	0	1	1	1	$\frac{A}{B}$	—	$\frac{D}{I}$	1	0	0	1	0	1	1	1	1
	07---57	AND	0	1	1	1	$\frac{A}{B}$	—	$\frac{D}{I}$	1	0	1	0	1	1	1	1	1
	07---677	DSZ	0	1	1	1	$\frac{A}{B}$	—	1	1	0	1	1	1	1	1	1	1
	07---717	JSM	0	1	1	1	$\frac{A}{B}$	—	1	1	1	0	0	1	1	1	1	1
	07---37	JMP	0	1	1	1	$\frac{A}{B}$	—	$\frac{D}{I}$	1	1	0	1	1	1	1	1	1

CONTINUED

CODING TABLE CONTINUED

GROUP	OCTAL	INSTR	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP EXECUTE DMA	07-016	EX*	0	1	1	1	$\frac{1}{2}$	-	-	-	-	1	0	0	1	1	1	0
	070036	DMA	0	1	1	1	0	-	-	-	-	0	1	1	1	1	1	0
	07-056	CM*	0	1	1	1	$\frac{1}{2}$	-	-	-	-	1	0	1	1	1	1	0
	07-076	TC*	0	1	1	1	$\frac{1}{2}$	-	-	-	-	1	1	1	1	1	1	0
INPUT OUTPUT GROUP	1727--	STF	1	1	1	1	-	1	0	1	1	1	SELECT CODE					
	1737--	CLF	1	1	1	1	-	1	1	1	1	1						
	17-7--	SFC	1	1	1	1	-	1	$\frac{1}{2}$	1	1	1						
	17-5--	SFS	1	1	1	1	-	1	$\frac{1}{2}$	1	0	1						
	17-5--	CLC	1	1	1	1	-	1	$\frac{1}{2}$	1	0	1						
	17-6--	STC	1	1	1	1	-	1	$\frac{1}{2}$	1	1	0						
	17-1--	OT*	1	1	1	1	$\frac{1}{2}$	1	$\frac{1}{2}$	0	0	1						
	17-2--	LI*	1	1	1	1	$\frac{1}{2}$	1	$\frac{1}{2}$	0	1	0						
	17-0--	MI*	1	1	1	1	$\frac{1}{2}$	1	$\frac{1}{2}$	0	0	0						
MAC GROUP	170402	RET	1	1	1	1	0	0	0	1	0	0	0	0	0	0	1	0
	170002	MOV	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0
	170000	CLK	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	170004	XFR	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	174430	MRX	1	1	1	1	0	0	0	1	0	0	0	1	1	0	0	0
	174470	MRY	1	1	1	1	1	0	0	1	0	0	1	1	1	0	0	0
	171400	MLS	1	1	1	1	0	0	0	1	0	0	1	1	1	0	0	0
	170410	DAS	1	1	1	1	0	0	0	1	0	0	0	0	1	0	0	0
	175400	DLS	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
	170560	FKA	1	1	1	1	0	0	0	1	1	0	1	1	1	0	0	0
	171460	FMP	1	1	1	1	0	0	0	1	1	0	1	1	1	0	0	0
	170420	FDV	1	1	1	1	0	0	0	1	1	0	0	0	1	0	0	0
	174400	CMX	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0
	170400	CMY	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0
	170540	NDI	1	1	1	1	0	0	0	1	0	1	1	0	0	0	0	0
	171450	NRM	1	1	1	1	0	0	1	1	0	0	1	0	1	0	0	0

DETAILED LISTING OF ROUTINES
AND SUBROUTINES OF BASIC INSTRUCTIONS

A complete listing of all of the routines and subroutines of basic instructions employed by the calculator and of all of the constants employed by these routines and subroutines is given below. All of these routines, subroutines, and constants are stored either in the basic ROM or in the plug-in ROM modules employed therewith. Each page within the listing is numbered at the upper left-hand corner. Each line of each page is separately numbered in the first column from the left-hand side of the page. This facilitates reference to different parts of the listing. Descriptive headings are also provided throughout the listing to identify routines, subroutines, groups of constants, different portions of the ROM, the plug-in ROM modules, etc. Each instruction of each routine or subroutine and each constant stored in the ROM or plug-in ROM modules is represented in octal form by six digits in the third column from the left-hand side of the page, and the address of the ROM location in which each such instruction or constant is stored is represented in octal form by five digits in the second column from the left-hand side of the page.

Mnemonic labels serving as symbolic addresses or names are given in the fourth column from the left-hand side of the page for most of the constants and many of the instructions to facilitate references to these constants and instructions and associated instructions. The mnemonic code of each basic instruction and of each pseudo instruction is given in the fifth column from the left-hand side of the

5 page. As noted above, each basic instruction is employed as a step in a routine or
subroutine of one or more basic instructions and therefore has an address in the
ROM. Pseudo instructions such as ORG, EQU, etc. which appear (and are
recognizable as not being one of the 71 basic machine instructions listed above)
are used for control of the Assembler, which translates the symbolic/mnemonic
coding of the fourth, fifth, and sixth columns into the address and contents of
ROM registers which appear in the second and third columns. (See chapter 4 of
the Hewlett-Packard "Assembler Programmer's Reference Manual" of April,
1970.) They are not employed as steps in the routines and subroutines performed
by the calculator and therefore have no addresses in the ROM. Mnemonic
operand codes are given in the sixth column from the left-hand side of the page,
and descriptive comments are given to the right of the sixth column. The format,
assembly, and use of the listing is explained in greater detail in the above-
mentioned Hewlett-Packard "Assembler Programmer's Reference Manual".

10 In addition, cross-reference symbol tables are included directly below each
group of routines and subroutines for providing alphabetical listing of each of the
mnemonic labels and operands. The number contained in the first column to the
right of each mnemonic in the cross-reference table represents the line in the
associated routine or subroutine listing at which that mnemonic appears as a label
in column four. Subsequent columns beyond the first column to the right of each
mnemonic in the cross-reference table represent the line in the associated routine
or subroutine listing at which that mnemonic appears as an operand in column six.

15

20



```
0001          ASMB,A,L,C
0003* MAIN SYSTEM FIRMWARE LISTING 3-27-72
0004*
0005 00000          ORG 0
0006*
0007 00000 164121    JMP START,I  START OF MONITOR
0008 00001 043060    OCT 43060    CONSTANT NEEDED BY PROCESSOR
0009 00002 161463    JSM INTEL,I  INTERRUPT
0010*
0011 00003 000000    C6          OCT 0          ONE
0012 00004 010000    OCT 010000    0000
0013 00005 000000    OCT 0          0000
0014*
0015 00006 000000    C5          OCT 000000    PI/2
0016 00007 012560    OCT 12560    1570
0017 00010 074543    OCT 074543    7963
0018 00011 023200    OCT 023200    2680
0019*
0020 00012 000001    .1          DEC 1
0021*
0022 00013 000400    C7          OCT 000400    180/PI
0023 00014 053451    OCT 053451    5729
0024 00015 053571    OCT 053571    5779
0025 00016 050450    OCT 050450    5128
0026*
0027 00017 000002    .2          DEC 2
0028 00020 000003    .3          DEC 3
0029 00021 000004    .4          DEC 4
0030 00022 000005    .5          DEC 5
0031 00023 000006    .6          DEC 6
0032 00024 000007    .7          DEC 7
0033 00025 000010    .8          DEC 8
0034 00026 000011    .9          DEC 9
0035 00027 000013    .11         DEC 11
0036 00030 000014    .12         DEC 12
0037 00031 000015    CRTN        OCT 15
0038 00031          EOL          EQU CRTN
0039 00031          .13         EQU CRTN
0040*****
0041* POINTERS TO FUNCTIONS CONSTANTS
0042*
0043 00027          DPERR        EQU .11
0044 00023          PI/2         EQU .6
0045 00020          ONE          EQU .3
0046*
0047*****
0048 00032 000017    .15         DEC 15
0049 00033 000020    .16         DEC 16
0050 00034 000025    .21         DEC 21
0051 00035 000026    .22         DEC 22
0052 00036 000027    .23         DEC 23
0053 00037 000034    .28         DEC 28
0054 00040 000037    .31         DEC 31
0055 00041 000040    .32         DEC 32
0056 00042 000041    .33         DEC 33
0057 00043 000042    .34         DEC 34

    POINTER TO 180/PI
    POINTER TO PI/2
    POINTER TO THE CONSTANT ONE
```

PAGE 2

9830 MAIN SYSTEM SOFTWARE

0058	00044	000045	.37	DEC	37
0059	00045	000050	.40	DEC	40
0060	00046	000051	.41	DEC	41
0061	00047	000053	.43	DEC	43
0062	00050	000055	.45	DEC	45
0063	00051	000056	.46	DEC	46
0064	00052	000060	.48	DEC	48
0065	00053	000072	.58	DEC	58
0066	00054	000077	.63	DEC	63
0067	00055	000111	I	OCT	111
0068	00056	023420	.1E4	DEC	10000
0069	00057	001750	.1000	DEC	1000
0070	00060	000144	.100	DEC	100
0071	00061	000012	.10	DEC	10
0072	00061		LNFD	EQU	.10
0073	00062	100000	FLGBT	OCT	100000
0074	00063	000143	.99	DEC	99
0075	00064	100004	INTFL	OCT	100004
0076	00065	076000	OPMSK	OCT	76000
0077	00066	077777	INF	OCT	77777
0078	00067	100037	TYPFL	OCT	100037
0079	00070	101777	OPDMK	OCT	101777
0080	00071	000105	E	OCT	105
0081	00072	000106	F	OCT	106
0082	00073	000123	S	OCT	123
0083	00032		B17	EQU	.15
0084	00033		B20	EQU	.16
0085	00074	000022	B22	OCT	22
0086	00075	000023	B23	OCT	23
0087	00035		B26	EQU	.22
0088	00036		B27	EQU	.23
0089	00076	000036	B36	OCT	36
0090	00040		B37	EQU	.31
0091	00043		B42	EQU	.34
0092	00077	000044	B44	OCT	44
0093	00100	000052	B52	OCT	52
0094	00053		B72	EQU	.58
0095	00101	000100	B100	OCT	100
0096	00102	000133	B133	OCT	133
0097	00103	000140	B140	OCT	140
0098	00104	000177	B177	OCT	177
0099	00105	000377	B377	OCT	377
0100	00013		B400	EQU	C7
0101	00106	000777	B777	OCT	777
0102	00107	001000	B1000	OCT	1000
0103	00110	001400	B1400	OCT	1400
0104	00111	001740	B1740	OCT	1740
0105	00112	001760	B1760	OCT	1760
0106	00113	001774	B1774	OCT	1774
0107	00114	001775	B1775	OCT	1775
0108	00115	001777	B1777	OCT	1777
0109	00116	002000	QTOP	OCT	2000
0110	00117	002400	B2400	OCT	2400
0111	00120	003000	B3000	OCT	3000
0112	00121	004000	B4000	OCT	4000
0113	00122	006000	B6K	OCT	6000

PAGE 3

9830 MAIN SYSTEM SOFTWARE

0114	00004	B10K	EQU	C6+1	
0115	00004	RPOP	EQU	B10K	
0116	00123	012000	OCT	12000	
0117	00123	B12K	OCT	12000	
0118	00123	RBOP	EQU	B12K	
0119	00124	032000	OCT	32000	
0120	00125	050000	OCT	50000	
0121	00126	052000	OCT	52000	
0122	00127	170000	OCT	170000	
0123	00130	013777	OCT	13777	
0124	00131	37777	OCT	37777B	LAST ROM BLOCK
0125	00132	17777	DEC	-1	
0126	00133	17776	DEC	-2	
0127	00134	17775	DEC	-3	
0128	00135	17774	DEC	-4	
0129	00136	17773	DEC	-5	
0130	00137	17772	DEC	-6	
0131	00140	17767	DEC	-9	
0132	00141	17766	DEC	-10	
0133	00142	17765	DEC	-11	
0134	00143	17764	DEC	-12	
0135	00144	17763	DEC	-13	
0136	00145	17762	DEC	-14	
0137	00146	17761	DEC	-15	
0138	00147	17751	DEC	-23	
0139	00150	17747	DEC	-25	
0140	00151	17740	DEC	-32	
0141	00152	17734	DEC	-36	
0142	00153	17720	DEC	-48	
0143	00154	17700	DEC	-64	
0144	00155	17660	DEC	-80	
0145	00156	17634	DEC	-100	
0146	00157	17400	DEC	-256	
0147	00160	176000	DEC	-1024	
0148	00161	154360	DEC	-10000	—(MAXIMUM ALLOWABLE LINE
0149*	00162	043116	ASC	1, FN	NUMBER)
0150*	MATH CONSTANTS THAT ARE NEEDED ON BASE PAGE				
0151*					
0152	00163	177401	OCT	177401	
0153	00164	063440	OCT	063440	
0154	00165	061400	OCT	061400	

PAGE 4

9830 MAIN SYSTEM SOFTWARE

0156*						
0157	00166	40440	FWAM	DEF	40440B	FIRST WORD R/W
0158	00122		RSTKA	EQU	B1760	START OF RETURN STACK
0159	00167	002454	WRITE	DEF	WRIT	PRINTER AND DISPLAY DRIVER
0160	00170	002533	CLERA	DEF	CLEAR	CLEAR REFRESH BUFFER
0161	00116		REED	EQU	QTOP	KEYBOARD DRIVER QTOP IS=2000B
0162	00121		START	EQU	B4000	
0163	00171	004741	RUNA	DEF	RUN	PHASE 2; SYMBOL TABLE
0164	00172	010012	XEC4A	DEF	XEC4	STATEMENT EXECUTION EXIT
0165	00173	004044	PEXMA	DEF	PEXMK	
0166	00174	004037	RDYDA	DEF	RDYPT	
0167	00122		LISTA	EQU	B6K	
0168	00175	003072	TSRHA	DEF	TBSRH	TABLE SEARCH
0169	00176	007450	ELINA	DEF	OLINE	OUTPUT I/O BUFFER
0170	00177	005305	SSYMA	DEF	SSYMT	
0171	00200	011433	OPCHA	DEF	OPCHK	CHECK FOR UNDEFINED OPERAND
0172	00201	010563	FORMA	DEF	FORMX	EVALUATE FORMULA
0173	00202	012040	FSCA	DEF	FSC	
0174	00203	004511	ACTSA	DEF	ACTST	
0175	00204	004500	DELSA	DEF	DELST	
0176	00205	004355	AERRA	DEF	AERR	
0177	00206	004360	ERRA	DEF	ERROR	
0178	00207	000000	INTEA	BSS	1	KEYBOARD INTERRUPT LINK
0179	00210	000000	OUTIA	BSS	1	OUTPUT AN INTEGER
0180	00211	003200	.BUFA	DEF	IOBUF+ IOBUF	FIRST CHAR OF I/O BUFFER
0181	00212	003322	.SLBF	DEF	RBUFF+ RBUFF	FIRST CHAR OF REFRESH BUFFER
0182	00213	001744	ARIA	DEF	AR1	
0183	00214	001754	AR2A	DEF	AR2	
0184	00215	001773	TMPOP	DEF	RES	RESULT REGISTER POINTER
0185	00216	000000	.FADA	BSS	1	ADD
0186	00217	000000	.FSBA	BSS	1	SUBTRACT
0187	00220	000000	.FMPA	BSS	1	MULTPY
0188	00221	000000	.FDVA	BSS	1	DIVIDE
0189	00222	005447	ENTFA	DEF	ENTFN	
0190	00223	000000	.FSR	BSS	1	SQUARE ROOT
0191	00224	000000	UPARA	BSS	1	UP ARROW
0192	00225	000000	FLTRA	BSS	1	
0193	00226	000000	FLTFA	BSS	1	
0194	00227	000000	FXFLA	BSS	1	
0195	00230	000000	FLTSA	BSS	1	
0196	00231	000000	SPLFA	BSS	1	
0197	00232	000000	SNFLA	BSS	1	
0198	00233	001551	BEND	DEF	RBUFF	I/O BUFFER END
0199	00233		SBUFA	EQU	BEND	
0200	00234	006353	NMOTA	DEF	NMOUT	NUMBER OUTPUT ROUTINE
0201	00235	007027	FILLA	DEF	FILL	OUTPUT BLANKS
0202	00236	003442	SLEND	DEF	RBUFF+ RBUFF+ 80	REFRESH BUFFER END
0203	00111		TPMA	EQU	B1740	
0204	00110		BADRP	EQU	B1400	POINTER POINTER TO I/O BUFFER
0205*						
0206	00123		FOPBS	EQU	B12K	
0207	00237	016516	SALTA	DEF	SALT	
0208	00240	005257	MDIMA	DEF	MDIM	
0209	00241	010413	SYSTA	DEF	SYSTH	
0210	00242	005711	DCOMA	DEF	DCOMP	
0211	00243	000000	NUMCA	BSS	1	

108

116

1,444,141

116

PAGE 5

9830 MAIN SYSTEM SOFTWARE

```
0212 00244 000000 NUMOA BSS 1
0213 00245 007656 GTSTA DEF GETST
0214 00246 012231 VAROA DEF VAROP
0215 00247 010112 ECALA DEF ECAL EXECUTE CALC MODE
0216 00250 004625 OVCHA DEF OVCHK
0217 00251 004575 CLPIA DEF CLPRI
0218 00252 005416 ENTEA DEF ENTER
0219 00253 007672 CHRSA DEF CHRST
0220 00254 003317 XSCRA DEF XSCR1 SCRATCH KEYS
0221 00255 010040 XEC6A DEF XEC6
0222 00256 010425 FDAAA DEF FDATA
0223 00257 005660 FNDAA DEF FNDAD
0224 00260 005671 FND1A DEF FND1
0225 00261 002547 IOSTA DEF IOST I/O STATUS CHECK
0226 00262 010467 INREA DEF INRET
0227 00263 003336 SUMPA DEF SUMP
0228 00264 002504 CRCKA DEF CRCHK OUTPUT SUPRESSED CRLF
0229 00265 016501 IOHEA DEF IOHED
0230 00266 007240 QTCHA DEF QTCHK CHECK FOR STRING
0231 *****
0232*
0233 00267 000000 BSS 1 CHECKSUM WORD*****
```

PAGE 6

9830—ERROR TABLE

0235	00270	000625	E1	DEF SYSER	ROM CONFIGURATION ERROR
0236	00271	000561		DEF FSCE4	MEMORY OVERFLOW
0237	00272	004177		DEF NCALC	STMT NOT ALLOWED IN CALC MODE
0238	00273	000726		DEF SYE25	MISSING OR BAD INTEGER
0239	00274	004243	E5	DEF NSTMT	NO STATEMENT OR COMMAND RECOGNISED
0240	00275	012134		DEF FSCE3	UNRECOGNISED OPERAND
0241	00276	001152		DEF NOEOF	CHARS FOLLOW STMT LOGICAL END
0242	00277	004307		DEF SYE15	MISSING PRINT DELIMETER
0243	00300	004253		DEF UKERR	IMPROPER TYPING AID ASSIGNMENT
0244	00301	003155	E10	DEF KERR	KEY UNDEFINED
0245	00302	000000		BSS 1	
0246	00303	000000		BSS 1	
0247	00304	001165		DEF SYE12	SIGN GIVEN BUT NO NUMBER
0248	00305	000511		DEF COMCE+1	MISSING COMA
0249	00306	000522	E15	DEF FSCE1	NO LEFT PAREN
0250	00307	000535		DEF FSCE2	NO RIGHT PAREN
0251	00310	016432		DEF ARRE1	MISSING SUBSCRIPT
0252	00311	012452		DEF STRNP	STRING NOT PERMITTED
0253	00312	007660		DEF NOST	MISSING OR BAD STRING
0254	00313	007665	E20	DEF SYE14	NO CLOSING QUOTE
0255	00314	012433		DEF SYNE4	MISSING OR BAD FUNCTION I.D.
0256	00315	012515		DEF SYNE5	MISSING PARAMETER
0257	00316	003504		DEF NDATA	MISSING OR BAD DATA ITEM
0258	00317	007723		DEF SYNE7	MISSING THEN
0259	00320	003565	E25	DEF MISOF	MISSING OF
0260	00321	016461		DEF SYE13	MISSING VARIABLE
0261	00322	012464		DEF SYNE8	MISSING FOR VARIABLE
0262	00323	012473		DEF SYNE9	MISSING TO
0263	00324	012500		DEF SYE10	MISSING STEP
0264	00325	012522	E30	DEF SYNE6	MISSING ASSIGNMENT IN FOR STMT
0265	00326	004273		DEF SYNE2	LET STMT HAS NO STORE
0266	00327	003531		DEF FMTER	IMPROPER FORMAT SPEC
0267	00330	016436		DEF SYNE3	MISPLACED COM STATEMENT
0268	00331	005067		DEF CMERR	IMPROPER COM DECLARATION
0269	00332	005037	E35	DEF MER5	ARRAY DOUBLY DEFINED
0270	00333	005372		DEF SSYM3	PRECISION DOUBLY DEFINED
0271	00334	005353		DEF SYME2	INCONSISTENT DIMENSIONS
0272	00335	005157		DEF MER10	ARRAY HAS UNKNOWN DIMENSIONS
0273	00336	005276		DEF MER9	DIMENSIONS TOO LARGE
0274	00337	011440	E40	DEF ER8	OPERAND IS UNDEFINED
0275	00340	005421		DEF NINIT	ARRAY OR STRING NOT INITIALIZED
0276	00341	011240		DEF ER6	SUBSCRIPT EXCEEDS BOUNDS
0277	00342	007372		DEF SCERR	SELECT CODE LESS THAN 1 OR GTR THAN 15
0278	00343	010224		DEF NGOTO	LINE NOT FOUND OR IMPROPER STMT FOUND
0279	00344	007064	E45	DEF TABER	STATEMENTS USING FORMAT CANNOT USE TAB
0280	00345	007414		DEF FNEST	STATEMENTS USING FORMAT CANNOT BE NESTED
0281	00346	010231		DEF ER3	IMPROPER RETURN
0282	00347	010335		DEF BERR	IMPROPER FOR LOOP
0283	00350	010444		DEF ER60	OUT OF DATA
0284	00351	010041	E50	DEF MER11	LAST STMT IS NOT "END"
0285	00352	000000		BSS 1	
0286	00353	000000		BSS 1	
0287	00354	000000		BSS 1	
0288	00355	000000		BSS 1	

118

1,444,141

118

PAGE 7

BASE PAGE ROUTINES

0290*

0291* MULTI EXIT POINT RETURNS

0292*

0293 00356 060365 RETN3 JSM RET2 RETURN TO P+3
0294 00357 031401 RETN2 STA RSAVE RETURN TO P+2

0295 00360 021777 LDA RSPTR

0296 00361 000132 ADA M1

0297 00362 070637 ISZ A,I INCREMENT RETURN ADDRESS

0298 00363 021401 LDA RSAVE RESTORE (A)

0299 00364 170402 RET

0300*

0301 00365 031401 RET2 STA RSAVE

0302 00366 021777 LDA RSPTR

0303 00367 000133 ADA M2

0304 00370 070637 ISZ A,I INCREMENT RETURN ADDRESS

0305 00371 021401 LDA RSAVE RESTORE (A)

0306 00372 170402 RET

0307*

0308 00373 024212 ICNT LDB SLBF GET SINGLE LINE CHARACTER COUNT

0309 00374 074117 WBUFF LDA B

0310 00375 070076 TCA

0311 00376 001400 ADA BADDR COMPUTE CHAR COUNT

0312 00377 170402 RET

0313*

0314* GET A PARAMETER IN (B) FROM P+1

0315*

0316 00400 025777 GPARM LDB RSPTR

0317 00401 004133 ADB M2

0318 00402 074637 ISZ B,I

0319 00403 074537 LDB B,I

0320 00404 065375 JMP FIX+4

0321*

0322* TRANSFER FROM AR2 to AR1

0323*

0324 00405 024213 XFAR1 LDB ARIA

0325 00406 020214 LDA AR2A

0326 00407 170004 XFR

0327 00410 004134 ADB M3 RESTORE (B)

0328 00411 170402 RET

0329*

0330* CONVERT AND STORE AN OPERAND VALUE

0331*

0332* AR2=SOURCE

0333* (A)=DESTINATION TYPE

0334* (B)=DESTINATION ADDRESS

0335*

0336 00412 125413 RESULT LDB HSTPT,I

0337 00413 045413 ISZ HSTPT

0338 00414 121413 LDA HSTPT,I

0339 00415 045413 ISZ HSTPT

0340 00416 070206 STORE RAR 5 IF BIT 4=1

0341 00417 071352 SAM XFAR1+1 IT IS FULL PRECISION

0342 00420 007062 SAR 13 NO, GET PRECISION BITS

0343 00421 071250 SZA XFAR1+1 FULL?

0344 00422 035727 STB MBIN2 SAVE ADDRESS

0345 00423 010012 CPA .1 NO, SPLIT?

PAGE 8

BASE PAGE ROUTINES

0346 00424 064441 JMP STOR2 YES
 0347 00425 060405 STOR1 JSM XFAR1 INTEGER: SAVE IN AR1
 0348 00426 160225 JSM FLTRA,I CONVERT INTEGER
 0349 00427 064432 JMP INTOV OVERFLOW EXIT
 0350 00430 135727 STB MBIN2,I STORE INTEGER
 0351 00431 064444 JMP XFAR2 RESTORE AR2
 0352 00432 160205 INTOV JSM AERRA,I EMIT OVERFLOW
 0353 00433 177627 DEC -105 INTEGER OVERFLOW
 0354 00434 024066 LDB INF SBT MAXIMUM POSITIVE INTEGER
 0355 00435 021744 LDA AR1 GET MANTISSA SIGN
 0356 00436 071511 SLA STOR1+3 POSITIVE?
 0357 00437 074076 TCB NO
 0358 00440 064430 JMP STOR1+3
 0359 00441 060405 STOR2 JSM XFAR1 SAVE IN AR1
 0360 00442 025727 LDB MBIN2 RECALL ADDRESS
 0361 00443 160230 JSM FLTSA,I CONVERT TO SPLIT AND STORE
 0362*
 0363 00444 020213 XFAR2 LDA ARIA
 0364 00445 024214 LDB AR2A
 0365 00446 170004 XFR
 0366 00447 170402 RET
 0367* STORE AN OPERAND NAME AND TYPE
 0368*
 0369* (A)=ASCII CODE FOR LETTER
 0370* ENTER AT STROP+1 WITH PRE-ADJUSTED OPERAND TYPE
 0371* EXIT* (A)=COMPLETED OPERATOR—OPERAND PAIR
 0372*
 0373 00450 004151 STROP ADB M32 NUMERICALLY ADJUST OPERAND TYPE
 0374 00451 000154 ADA M64 GET 5 BIT CODE FOR OPERAND NAME
 0375 00452 070544 SAL 5 COMBINE THE
 0376 00453 074217 IOR B TWO OPERAND PARTS AND THE
 0377 00454 141706 IOR SBPTR,I OPERATOR—OPERAND PAIR
 0378*
 0379* UPDATE SYNTAX BUFFER POINTER AND CHECK FOR OVERFLOW
 0380*
 0381 00455 131706 STA SBPTR,I STORE (A) BEFORE UPDATE
 0382 00456 0045706 SBPUD ISZ SBPTR UPDATE POINTER
 0383 00457 131706 STA SBPTR,I SAVE (A)
 0384 00460 170402 RET
 0385*
 0386* FIND AND STORE ONE—CHARACTER OPERATORS
 0387* (A)=SOURCE CHARACTER
 0388* (B)=NUMBER OF ENTRIES TO BE SEARCHED
 0389* P+1=STARTING TABLE ADDRESS-1
 0390*
 0391 00461 024017 SYMC2 LDB .2 ALTERNATE ENTRY TO SEARCH TWO
 SYMBOLS
 0392 00462 064464 JMP SYMCK
 0393 00463 024012 SYMC1 LDB .1 ALTERNATE ENTRY TO SEARCH ONE
 SYMBOL
 0394*
 0395 00464 035466 SYMCK STB LOCL1 SAVE # TO BE SEARCHED
 0396 00465 031467 STA LOCL2 SAVE SOURCE CHARACTER
 0397 00466 060400 JSM GPARM GET STARTING ADDRESS
 0398 00467 074070 SYM1 SIB *+1 INCREMENT TABLE ADDRESS
 0399 00470 074517 LDA B,I GET TABLE ENTRY
 0400 00471 070342 SAR 8 POSITION IT
 0401 00472 011467 CPA LOCL2 MATCH SOURCE CHAR?

PAGE 9

BASE PAGE ROUTINES

0402	00473	064500		JMP	SYM2	YES
0403	00474	055466		DSZ	LOCL1	NO, CONTINUE SEARCH?
0404	00475	064467		JMP	SYM1	YES
0405	00476	021467		LDA	LOCL2	NO, RESTORE CHARACTER
0406	00477	170402		RET		RETURN TO P+2- NOT FOUND
0407	00500	020123	SYM2	LDA	FOPBS	COMPUTE
0408	00501	07076		TCA		
0409	00502	074017		ADA	B	OP-CODE
0410	00503	064543		JMP	STSRH+4	SHIFT-STORE-RETURN TO P+3
0411*						
0412*	CHECK FOR A COMMA					
0413*						
0414	00504	060463	COMCK	JSM	SYMC1	CHECK FOR
0415	00505	012001		DEF	12001B	COMMA-1
0416	00506	170402		RET		NOT FOUND — RETURN TO P+1
0417	00507	064357		JMP	RETN2	FOUND — RETURN TO P+2
0418*						
0419	00510	060504	COMCE	JSM	COMCK	CHECK FOR COMMA
0420	00511	160206		JSM	ERRA,I	ERROR IF NOT FOUND
0421	00512	170402		RET		
0422*						
0423*	CHECK FOR LEFT PARENTHESIS					
0424*						
0425	00513	060461	LPCK	JSM	SYMC2	LEFT PAREN
0426	00514	012023		DEF	12023B	LBRAC-1
0427	00515	170402		RET		NO, RETURN TO P+1
0428	00516	020126		LDA	LPOP	YES, RECODE A LEFT
0429	00517	064544		JMP	STSRH+5	PAREN AND RETURN TO P+2
0430*						
0431	00520	061055		JSM	GNEXT	GET NEXT CHAR
0432	00521	060513	LPCKE	JSM	LPCK	DEMAND A
0433	00522	160206	FSCE1	JSM	ERRA,I	LEFT PAREN OR BRACKET
0434	00523	170402		RET		
0435*						
0436*	CHECK FOR RIGHT PARENTHESIS					
0437*						
0438	00524	060461	RPCK	JSM	SYMC2	RIGHT PAREN
0439	00525	012003		DEF	12003B	PRAN-1
0440	00526	170402		RET		NO, RETURN TO P+1
0441	00527	020004		LDA	RPOP	YES, RECODE A RIGHT
0442	00530	060455		JSM	SBPUD-1	PAREN AND INC SBPTR
0443	00531	061055		JSM	GNEXT	GET NEXT CHAR
0444	00532	064357		JMP	RETN2	RETURN TO P+2
0445*						
0446	00533	061055		JSM	GNEXT	GET NEXT CHAR
0447	00534	060524	RPCKE	JSM	RPCK	DEMAND A RIGHT
0448	00535	160206	FSCE2	JSM	ERRA,I	LEFT PAREN OR BRACKET
0449	00536	170402		RET		
0450						
0451*	SEARCH TABLE AND STORE OP CODE IF FOUND					
0452*						
0453	00537	160175	STSRH	JSM	TSRHA,I	SEARCH TABLE
0454	00540	170402		RET		NOT FOUND
0455	00541	031466		STA	LOCL1	SAVE OP CODE
0456	00542	050040		AND	B37	BET 5 BIT CODE
0457	00543	070304		SAL	10	SHIFT AND

PAGE 10

BASE PAGE ROUTINES

0458	00544	131706		STA	SBPTR,I	STORE OP CODE
0459	00545	064357		JMP	RETN2	RETURN TO P+2
0460*						
0461*	SEARCH TABLE AND STORE EXPANDED OP CODE					
0462*				JSM	TSRHA,I	SEARCH TABLE
0463	00546	160175	STEXP	RET		
0464	00547	170402		SAL	5	ASSEMBLE AND
0465	00550	070544		IOR	OPMSK	STORE
0466	00551	040065		JMP	STSRH+5	
0467	00552	064544				
0468*						
0469*	BUMP H—STACK POINTER AND CHECK FOR OVERFLOW					
0470*				DSZ	HSTPT	DINC POINTER
0471	00553	055413	BHSTP	LDB	HSTPT	
0472	00554	025413		TCB		
0473	00555	074076		ADB	LSTPT	
0474	00556	005412		ADB	.2	
0475	00557	004017		SBM	STSRH+1	OVERLAP? RETURN IF NOT
0476	00560	075012		JSM	ERRA,I	
0477	00561	160206	MERB			
0478*						
0479	00561		FSCE4	EQU	MER8	
0480*						
0481*	STACK (B) ON L—STACK AND CHECK OVERFLOW					
0482*				LDB	PRADD	
0483	00562	025421		ISZ	LSTPT	INC POINTER
0484	00563	045412	SLWST	STB	LSTPT,I	STORE VALUE
0485	00564	135412		JMP	BHSTP+1	CHECK FOR OVERFLOW
0486	00565	064554				
0487*						
0488*	FETCH TOP OF L—STACK					
0489*				STA	PRADD	
0490	00566	031421		LDA	LSTPT,I	
0491	00567	121412	UNSTK	DSZ	LSTPT	
0492	00570	055412		RET		
0493	00571	170402				
0494*						
0495*	PREPARE FOR BRANCH THROUGH TABLE					
0496*				LDB	PRADD	
0497	00572	025421		STB	TEMPS	
0498	00573	035472		LDB	TEMPS,I	READ AND SAVE
0499	00574	125472	NEXTL	SBP	*+1,C	CLEAR SECURITY BIT
0500	00575	074073		STB	.LNUM	CURRENT LINE NUMBER
0501	00576	035427		LDB	TEMPS	FIND ADDRESS
0502	00577	025472		JSM	NEXTA	OF NEXT STATEMENT
0503	00600	061007		STB	PRADD	SAVE IT
0504	00601	035421		ISZ	TEMPS	
0505	00602	045472		LDB	TEMPS,I	READ OP CODE
0506	00603	125472		SBP	*+1,C	
0507	00604	074073		ISZ	TEMPS	
0508	00605	045472		SBR	10	
0509	00606	074442	NEXTI	TCB		GET NEGATIVE OF OP CODE
0510	00607	074076		STB	LOCLI	SAVE—(OP CODE)
0511	00610	035466		LDA	TEMPS,I	
0512	00611	121472		AND	OPMSK	GET PAGE
0513	00612	050065				

PAGE 11

BASE PAGE ROUTINES

```

0514 00613 024130 FPAGE LDB FROMA GET FIRST TABLE ADDRESS
0515 00614 074457 CPA B,I IS THIS THE PAGE?
0516 00615 170402 RET YES, (B) = LAST WORD OF PAGE
0517 00616 014131 CPB LROMA ALL ROM'S SEARCHED?
0518 00617 064622 JMP FPAGI YES, CHECK BINARY AREA
0519 00620 004116 ADB QTOP NO, CHECK NEXT ROM
0520 00621 064614 JMP FPAGE+1
0521 00622 025403 FPAGI LDB MAW
0522 00623 111403 CPA MAW,I IS CODE IN R/W?
0523 00624 170402 RET YES, (B) = MAW
0524 00625 160206 SYSER JSM ERRA,I NO, THE PROPER BLOCK IS NOT IN
0525*
0526*
0527*
0528* SAVE BADDR AND SBPTR
0529*
0530 00626 025400 SAVBP LDB BADDR
0531 00627 035715 STB SBADR
0532 00630 025706 LDB SBPTR
0533 00631 035716 STB SSBPT
0534 00632 170402 RET
0535*
0536* RESTORE BADDR AND SBPTR
0537*
0538 00633 021715 RESB1 LDA SBADR
0539 00634 031400 STA BADDR
0540 00635 021716 LDA SSBPT
0541 00636 031706 STA SBPTR
0542 00637 170402 RET
0543*
0544 00640 060633 RESBP JSM RESB1
0545 00641 121706 LDA SBPTR,I ERASE
0546 00642 050065 AND OPMSK OPERAND
0547 00643 131706 STA SBPTR,I
0548 00644 170402 RET
0549*
0550* SCRATCH SYMBOL TABLE AND INITIALIZE STACKS
0551*
0552 00645 025404 MINIT LDB LWAM
0553 00646 035407 STB SYMTP
0554 00647 035406 STB VALUE
0555 00650 064653 JMP INITS+2
0556*
0557* INITIALIZE STACKS FOR SYNTAX
0558*
0559 00651 020662 INITS LDA HFLGA
0560 00652 170000 CLR CLEAR HALT AND TRACE FLAGS
0561 00653 025426 LDB PBPTO SCRATCH L-STACK
0562 00654 004132 ADB M1
0563 00655 035412 STB LSTPT
0564 00656 025407 INITI LDB SYMTP SCRATCH H-STACK
0565 00657 074070 SIB *+1
0566 00660 035413 STB HSTPT
0567 00661 164242 JMP DCOMA,I RESET DATA POINTERS
0568 00662 001436 HFLGA DEF HFLG1
0569*

```

135

PAGE 12

BASE PAGE ROUTINES

0570* SUBROUTINE TO BUILD A PROGRAM INTEGER LESS THAN 10000
0571*
0572 00663 055400 DSZ BADDR BACK UP POINTER
0573 00664 024161 LNUM LDB MAXSN CALL PGINT WITH —10000
0574 00665 145706 ISZ SBPTR,I FLAG FOR RESEQUENCABLE INTEGER
0575*
0576* SUBROUTINE TO BUILD A PROGRAM INTEGER AND STORE IT
0577* IN THE S-BUFFER AND FLAG THE PREVIOUS WORD
0578* (B) = —(MAXIMUM ALLOWABLE INTEGER)
0579*
0580 00666 121706 PGINT LDA SBPTR,I SET
0581 00667 072053 SAP *+1,S "INTEGER
0582 00670 000020 ADA 3 "FOLLOWS"
0583 00671 060455 JSM SBPUD-1 FLAG
0584 00672 035477 STB TEMP4 SAVE UPPER LIMIT
0585 00673 061055 JSM GNEXT
0586 00674 064677 JMP INTCK+1
0587*
0588* SUBROUTINE TO BUILD AN INTEGER AND STORE IT IN SBUFFER
0589* (A)=FIRST CHARACTER
0590* (B) = —(MAXIMUM ALLOWABLE INTEGER)
0591*
0592 00675 024161 LDB MAXSN BUILD A LINE #
0593*
0594 00676 035477 INTCK STB TEMP4 SAVE UPPER LIMIT
0595 00677 074742 SBR 16 INITIALIZE
0596 00700 035722 STB INTGR PARTIAL RESULT
0597 00701 060750 JSM DIGCK 1ST CHAR A DIGIT?
0598 00702 064725 JMP SYE25-1 NO
0599 00703 060750 JSM DIGCK DIGIT?
0600 00704 064716 JMP INTC2 NO
0601 00705 021722 LDA INTGR RECALL PARTIAL RESULT
0602 00706 035722 STB INTGR SAVE LATEST DIGIT
0603 00707 024061 LDB 10 MULTIPLY PARTIAL
0604 00710 061201 JSM IMPY RESULT BY 10
0605 00711 001722 ADA INTGR ADD LATEST DIGIT
0606 00712 070552 SAM SYE25-1 SKIP IF OVERFLOW
0607 00713 031722 STA INTGR STORE PARTIAL RESULT
0608 00714 061055 JSM GNEXT GET NEXT CHARACTER
0609 00715 064703 JMP INTC1
0610 00716 025722 INTC2 LDB INTGR SKIP THROUGH LINK
0611 00717 135706 STB SBPTR,I STORE INTEGER
0612 00720 074250 SZB SYE25-1 IF ZERO
0613 00721 005477 ADB TEMP4 SUBTRACT LIMIT
0614 00722 076153 SBP SYE25-1,S SKIP IF TOO LARGE
0615 00723 025722 LDB INTGR RETURN WITH
0616 00724 064456 JMP SBPUD INTEGER IN (B); NEXT CHAR IN (A)
0617 00725 045724 ISZ GFLAG RETURN ON ERROR FLAG SET?
0618 00726 160206 SYE25 JSM ERRA,I NO
0619 00727 064357 JMP RETN2 YES, RETURN TO P+2
0620*
0621* GET ADDRESSES FOR SYSTEM COMMANDS
0622*
0623* RETURN: TEMP4= ADDRESS OF FIRST LINE
0624* (B)= ADDRESS OF LAST LINE+1
0625*

PAGE 13

BASE PAGE ROUTINES

```

0626 00730 074070 GETAD SIB *+1 ADD 1 TO SECOND TO
0627 00731 035477 STB TEMP4 MAKE ADDRESS INCLUSIVE
0628 00732 074076 TCB IS FIRST
0629 00733 070037 ADB A GREATER
0630 00734 075513 SBP SYE25 THAN SECOND
0631 00735 061075 JSM FNDPS FIND ADDRESS OF FIRST
0632 00736 064740 JMP *+2
0633 00737 064740 JMP *+1
0634*
0635* CALL FNDPS AND RETURN TO P+1
0636*
0637 00740 021477 LDA TEMP4
0638 00741 035477 STB TEMP4
0639 00742 061075 FND JSM FNDPS
0640 00743 170402 RET
0641 00744 170402 RET
0642 00745 170402 RET
0643*
0644* CHECK IF CHAR IN (A) IS A DIGIT
0645* RETURN TO P+1 IF NOT FOUND
0646* RETURN TO P+2 IF FOUND; ASCII IN (A); DIGIT IN (B)
0647*
0648 00746 030071 DLIMIT OCT 30071 0-9
0649 00747 040532 LLIMIT OCT 40532 A-Z
0650 00750 024746 DIGCK LDB DLIMIT LODA LIMITS FOR DIGITS
0651 00751 064753 JMP LIMIT CHECK LIMITS AND RETRUN
0652*
0653* CHECK IF CHAR IN (A) IS A LETTER
0654* RETURN TO P+1 IF NOT FOUND
0655* RETURN TOP+2 IF FOUND
0656*
0657 00752 024747 LETCK LDB LLIMIT LOAD LIMITS FOR LETTER
0658* ENTER INTO LIMIT DIRECTLY
0659*
0660 00753 035466 LIMIT STB LOCL1 SAVE LIMITS
0661 00754 074346 RBR 8
0662 00755 074342 SBR 8 GET UPPER LIMITS
0663 00756 074056 CMB
0664 00757 070037 ADB A CODE EXCEEDS
0665 00760 075253 SBP FND+3 UPPER LIMIT?
0666 00761 025466 LDB LOCL1 NO
0667 00762 074342 SBR 8 GET LOWER LIMIT
0668 00763 074076 TCB
0669 00764 070037 ADB A CODE LESS THAN
0670 00765 075012 SBM FND+3 LOWER LIMIT
0671 00766 070137 LDB A NO, CODE IN LIMITS
0672 00767 004153 ADB M48 YES, RETURN TO P+2
0673 00770 064357 JMP RETN2
0674*
0675* GET OPERAND
0676*
0677 00771 121472 GTOPP LDA TEMPS,I
0678 00772 050070 AND OPDMK
0679 00773 170402 RET
0680***
0681*

```

PAGE 14

BASE PAGE ROUTINES

0682 00774 000000 BSS 10

0683*

0684***

0685*

0686*

0687*

0688*

COMPUTE ADDRESS OF NEXT LINE

0689*

0690*

01006 035466

STB LOCL1

0691*

01007 074117

NEXTA

LDA B

SAVE ADDRESS OF CURRENT LINE

0692*

01010 070070

SIA *+1

GET

0693*

01011 070517

LDA A,I

STATEMENT

0694*

01012 050104

AND B177

LENGTH

0695*

01013 070037

ADB A

COMPUTE ADDRESS OF NEXT LINE

0696*

01014 170402

RET

0697*

0698*

0699*

0700*

0701*

0702*

0703*

0704*

0705*

0706*

0707*

0708*

0709*

0710*

0711*

0712*

0713*

0714*

0715*

01015 020041

OBLNK

LDA 32

0716*

01016 024110

ONEXT

LDB BADRP

OPTIONAL ENTRY POINT

0717*

01017 031467

OUTCR

STA LOCL2

SAVE CHARACTER TO BE STORED

0718*

01020 074517

LDA B,I

FETCH BUFFER POINTER

0719*

01021 011623

CPA PEND

AT I/O FND?

0720*

01022 170402

RET

YES

0721*

01023 074637

ISZ B,I

BUMP BUFFER POINTER

0722*

01024 070137

LDB A

0723*

01025 074006

RBR 1

0724*

01026 070002

SAR 1

0725*

01027 070517

LDA A,I

FETCH BUFFER WORD

0726*

01030 074112

SBM *+2

0727*

01031 070346

RAR 8

SET A TO STORE IN LEFT HALF

0728*

01032 050157

AND M256

0729*

01033 041467

IOR LOCL2

0730*

01034 074132

SBM *+2,C

0731*

01035 070346

RAR 8

ROTATE IF FLAG WAS SET

0732*

01036 074557

STA B,I

STORE NEW BUFFER WORD

0733*

01037 170402

RET

0734*

0735*

0736*

0737*

THIS SUBROUTINE FETCHES
A CHARACTER FROM A BUFFER

(128)

126

1,444,141

126

PAGE 15

BASE PAGE ROUTINES

```

0738*
0739*
0740*   ENTRY CONDITIONS:
0741*   (B)=BUFFER POINTER POINTER
0742*
0743*
0744*   EXIT CONDITIONS:
0745*   (A)=8-BIT CHARACTER
0746*   (B)=BUFFER POINTER POINTER
0747*
0748*
0749*
0750  01040  024110  LDB  BADRP  GET NEXT CHAR FROM I/O BUFFER
0751  01041  035466  GETCH STB  LOCL1  SAVE POINTER POINTER
0752  01042  074517  LDA  B,I   FETCH BUFFER POINTER
0753  01043  070002  SAR  1
0754  01044  070517  LDA  A,I   FETCH TWO CHARACTERS
0755  01045  074537  LDB  B,I
0756  01046  074111  SLB  GETI  TOP HALF WORD?
0757  01047  070346  RAR  8     NO, MOVE RIGHT HLAF TO LEFT
0758  01050  070342  GETI SAR  8     TAKE TOP HALF
0759  01051  145466  ISZ  LOCL1,I BUMP POINTER
0760  01052  170402  RET                      YES
0761*
0762  01053  024111  GTMP LDB  TMPA
0763  01054  065041  JMP  GETCH
0764*
0765*
0766  01055  024110  GNEXT LDB  BADRP  GET CHAR. AND CONVERT
0767  01056  061362  JSM  GTCON
0768  01057  011707  CPA  BLANK  BLANK?
0769  01060  065055  JMP  *-3    YES, IGNORE IT
0770  01061  170402  RET                      YES, RETURN TO P+1
0771*
0772*   GET CHARACTER ROUTINES
0773*
0774  01062  020233  GFRST LDA  SBUFA
0775  01063  031706  STA  SBPTR  INITIALIZE SYNTAX BUFFER POINTER
0776  01064  020211  LDA  BUFA   SET BUFFER POINTER TO START
0777  01065  031400  STA  BADDR  OF I/O BUFFER
0778  01066  061055  GETSK JSM  GNEXT GET A CHARACTER
0779  01067  010031  CPA  EOL    END OF LINE?
0780  01070  170402  RET                      YES, RETURN TO P+1
0781  01071  064357  JMP  RETN2  NO, RETURN TO P+2
0782*
0783*   FIND SEQUENTIAL POSITION OF LINE # GIVEN IN (A)
0784*
0785*   RETURN P+1:BEYOND END OF PROGRAM
0786*   P+2:BEYOND LINES
0787*   P+3:FOUND
0788*   TEMP3=(B)=ADDRESS OF LINE OR NEXT LINE IN SEQUENCE
0789*   LSTLN=LINE NUMBER OF PREVIOUS LINE
0790*
0791  01072  120233  LDA  SBUFA,I GET SYNTAX SEQUENCE #
0792  01073  031431  STA  CLINE  SAVE FOR LIST ROUTINE
0793  01074  031427  STA  LNUM

```

PAGE 16

BASE PAGE ROUTINES

0794	01075	031476	FNDPS	STA	TEMP3	SAVE SEQUENCE #
0795	01076	025424		LDB	PBUFF	
0796	01077	015423	FNDP1	CPB	PBPTR	END OF PROGRAM?
0797	01100	065117		JMP	FNDP4	YES, RETURN TO P+1
0798	01101	074517		LDA	B,I	GET CURRENT #
0799	01102	070073		SAP	*+1,C	CLEAR SECURE BIT
0800	01103	070510		SZA	FNDP2	EXIT P+3 IF TYPING AID KEY
0801	01104	070076		TCA		SUBTRACT CURRENT # FROM
0802	01105	001476		ADA	TEMP3	DESIRED #
0803	01106	070412		SAM	FNDP3	CURRENT>DESIRED? IF YES, P+2
0804	01107	070330		SIA	FNDP2	EQUAL? IF YES, P+3 (A)=1
0805	01110	074517		LDA	B,I	SAVE
0806	01111	070073		SAP	*+1,C	
0807	01112	031720		STA	LASTN	CURRENT LINE
0808	01113	061007		JSM	NEXTA	COMPUTE ADDRESS OF NEXT LINE
0809	01114	065077		JMP	FNDP1	
0810	01115	060365	FNDP2	JSM	RET2	P+3
0811	01116	060365	FNDP3	JSM	RET2	P+2
0812	01117	035476	FNDP4	STB	TEMP3	P+1, SAVE STATEMENT ADDRESS
0813	01120	170402		RET		
0814*						
0815*						MOVE WORDS TO HIGHER MEMORY
0816*						
0817*						TEMP2=LAST WORD+1 OF SOURCE
0818*						TEMP3=FIRST WORD OF SOURCE
0819*						TEMP4=LAST WORD+1 OF DESTINATION
0820*						
0821	01121	025475	MVTOH	LDB	TEMP2	FETCH SOURCE ADDRESS
0822	01122	015476		CPB	TEMP3	ALL RELOCATION DONE?
0823	01123	170402		RET		
0824	01124	055477		DSZ	TEMP4	BACK UP DESTINATION POINTER
0825	01125	004132		ADB	M1	BACK UP SOURCE POINTER
0826	01126	074517		LDA	B,I	MOVE
0827	01127	131477		STA	TEMP4,I	WORD
0828	01130	065122		JMP	MVTOH+1	
0829*						
0830*						FETCH AN INTEGER ALLOWING ZERO
0831*						
0832	01131	020132	PGIN0	LDA	M1	FLAG TO RETURN
0833	01132	031724		STA	GFLAG	ON OVERFLOW OR 0
0834	01133	060666		JSM	PGINT	FETCH INTEGER
0835	01134	170402		RET		WITHIN LIMITS
0836	01135	045706		ISZ	SBPTR	OVERFLOW OR ZERO; UPDATE POINTER
0837	01136	075710		SZB	*-2	SKIP IF ZERO
0838	01137	064726		JMP	SYE25	MUST BE OVERFLOW
0839*						
0840*						FETCH A LETTER
0841*						EXIT: P+1 IF NEXT CHAR IS NOT A LETTER; CHAR IN (A)
0842*						P+2 IF LETTER; LETTER IN TEMP1; NEXT CHAR IN (A) AND TEMP2
0843*						
0844	01140	061055	LTR	JSM	GNEXT	
0845	01141	060752		JSM	LETCK	LETTER?
0846	01142	170402		RET		NO, RETURN TO P+1
0847	01143	031474		STA	TEMP1	YES, SAVE IT
0848	01144	061055		JSM	GNEXT	
0849	01145	031475		STA	TEMP2	SAVE SECOND CHARACTER

PAGE 17

BASE PAGE ROUTINES

```

0850 01146 064357 JMP RETN2 RETURN TO P+2
0851*
0852* END AND RESTORE STATEMENT SYNTAX
0853*
0854 01147 061055 ENDS JSM GNEXT CHECK FOR ADDITIONAL CHARACTERS
0855 01150 045706 ISZ SBPTR
0856 01151 061153 EOST JSM EOLCK
0857 01152 160206 NOEOF JSM ERR,A,I
0858*
0859* SUBROUTINE TO CHECK FOR EOL
0860*
0861 01153 010031 EOLCK CPA EOL EOL?
0862 01154 164203 JMP ACTSA,I YES, JUMP DIRECTLY TO ACTST
0863 01155 170402 RET
0864*
0865* INPUT A CONSTANT
0866*
0867 01156 061166 CONST JSM GSIGN GET SIGN
0868 01157 035721 STB SIGN STORE IT
0869 01160 160243 JSM NUMCA,I FETCH UNSIGNED NUMBER
0870 01161 065163 JMP CONS3 NONE FOUND
0871 01162 064357 JMP RETN2 FOUND, RETURN TO P+2
0872 01163 025721 CONS3 LDB SIGN
0873 01164 074610 SZB GSIGR SKIP IF NO SIGN GIVEN
0874 01165 160206 SYE12 JSM ERR,A,I SIGN GIVEN WITHOUT NUMBER
0875*
0876* GET THE SIGN OF A NUMBER
0877*
0878 01166 061055 GSIGN JSM GNEXT
0879 01167 074742 SBR 16
0880 01170 010047 CPA .43 "+"?
0881 01171 074230 SIB GSIGI+1 YES
0882 01172 010050 CPA .45 "-"?
0883 01173 024132 LDB M1 YES
0884 01174 074210 GSIG1 SZB GSIGR SKIP IF NO SIGN GIVEN
0885 01175 035470 STB L2T1 SAVE SIGN
0886 01176 061055 JSM GNEXT GET CHARACTER FOLLOWING SIGN
0887 01177 025470 LDB L2T1 RECALL SIGN
0888 01200 170402 GSIGR RET
0889*
0890* MULTIPLY 2 INTEGERS IN (A) AND (B) AND RETURN
0891* RESULT IN (A); IF PRODUCT > 32K, (A)<0
0892*
0893 01201 031466 IMPY STA MCAND SAVE MULTIPLICAND
0894 01202 070742 SAR 16 INITIALIZE PRODUCT
0895 01203 074151 IMPY1 SLB *+3 SKIP IF MULTIPLIER BIT IS ZERO
0896 01204 001466 ADA MCAND ADD MULTIPLICAND TO PRODUCT
0897 01205 071552 SAM IMPYR SKIP IF RESULT >32K
0898 01206 074002 SBR 1 SHIFT MULTIPLIER RIGHT
0899 01207 075450 SZB IMPYR RETURN IF FINISHED
0900 01210 031467 STA PROD SAVE PRODUCT
0901 01211 021466 LDA MCAND MULTIPLY
0902 01212 001466 ADA MCAND MULTIPLICAND
0903 01213 071252 SAM IMPYR SKIP IF RESULT >32K
0904 01214 031466 STA MCAND BY 2
0905 01215 021467 LDA PROD RECALL PRODUCT

```

PAGE 18

BASE PAGE ROUTINES

```

0906 01216 065203 JMP IMPY1
0907 01200 IMPYR EQU GSGR
0908*
0909* EVALUATE FORMULA AND FETCH RESULT TO AR2
0910*
0911 01217 160201 FETCH JSM FORMAI EVALUATE FORMULA
0912 01220 160200 JSM OPCHA,I GET ADDRESS OF RESULT
0913 01221 045413 ISZ HSTPT
0914 01222 045413 ISZ HSTPT REMOVE FROM STACK
0915 01223 074117 LDA B
0916 01224 064445 JMP XFAR2+1 TRANSFER TO AR2
0917*
0918* SUBROUTINE TO LEAVE KEY DEFINITION MODE
0919*
0920 01225 021422 LDEF LDA UKPTR
0921 01226 070410 SZA *+8
0922 01227 021425 LDA PBUFO
0923 01230 031424 STA PBUFF RESTORE PBUFF
0924 01231 021426 LDA PBPTO
0925 01232 031423 STA PBPTR RESTORE PBPTR
0926 01233 070742 SAR 16
0927 01234 031422 STA UKPTR CLEAR USER KEY FLAG
0928 01235 031432 STA DRQST CLEAR DATA REQUEST FLAG
0929 01236 170402 RET
0930*
0931*
0932* *****
0933*
0934* MATH ROUTINES
0935*
0936*
0937* STORE ROUTINES
0938*
0939 01237 074117 STAR2 LDA B ADD THE EXPONENT OF
0940 01240 001744 ADA AR1E AR1 AND THE B REGISTER,
0941 01241 050163 AND ZAP CLEAR OUT BITS 1-7 AND
0942 01242 031754 STA AR2E STORE IN AR2
0943*
0944 01243 074040 ABR 2 OFFSET THE TWO ABOVE
0945 01244 021744 LDA AR1E QUANTITIES AND RE-ADD
0946 01245 070040 AAR 2 TO CHECK FOR A
0947 01246 074017 ADA B MAXIMUM EXPONENT
0948 01247 070137 LDB A SAVE THE SIGN OF THE EXPONENT
0949 01250 070113 SAP *+2 IF IT IS POSITIVE
0950 01251 070076 TCA COMPLEMENT IT
0951 01252 000164 ADA MASK1 ADD A POSITIVE 100 TO IT
0952 01253 070312 SAM OF SKIP IF AN OVERFLOW HAS OCCURED
0953*
0954 01254 025755 ZONK LDB AR2E+1
0955 01255 076150 RZB ZONK1
0956 01256 020214 CLAR2 LDA AR2A CLEAR AR2
0957 01257 170000 CLR
0958 01260 170402 ZONK1 RET
0959*
0960*
0961 01261 021754 OF LDA AR2E OTHERWISE SET UP BITS 0 AND 15 OF B

```

PAGE 19

BASE PAGE ROUTINES

0962	01262	070111		SLA	*+2		
0963	01263	076051		SLB	*+1,S		FOR MAX OR MIN ANSWER AND ITS SIGN
0964*							
0965	01264	074213		SBP	OVFW		
0966*							
0967	01265	160205	UNFW	JSM	AERRA,I		
0968	01266	177633		DEC	-101		
0969	01267	065256		JMP	CLAR2		
0970*							
0971	01270	061274	OVFW	JSM	STMAX		
0972	01271	160205		JSM	AERRA,I		
0973	01272	177634		DEC	-100		
0974	01273	170402		RET			
0975*							
0976	01274	061256	STMAX	JSM	CLAR2		
0977	01275	074352		SBM	OVER		
0978*							
0979	01276	045757		ISZ	AR2E+3		
0980	01277	170400		CMY			
0981	01300	020165		LDA	MASK5		
0982	01301	031754		STA	AR2E		
0983	01302	074111		SLB	*+2		
0984	01303	045754		ISZ	AR2E		
0985	01304	170402	OVER	RET			
0986*							
0987	01305	070742	ROUND	SAR	16		
0988	01306	174470		MRY			
0989	01307	000136		ADA	M5		
0990	01310	070175		SEC	*+3,C		
0991	01311	020214		LDA	AR2A		
0992	01312	170540		MDI			
0993	01313	170402		RET			
0994*							
0995*							
0996*	*****						
0997*							
0998*							
0999*							
1000	01314	171400	ODGIT	MLS			
1001	01315	000052		ADA	.48		
1002	01316	065016		JMP	ONEXT		
1003*							
1004	01317	070537	SGNS	LDB	A,I		GET EXPONENT WORD
1005	01320	070070		SIA	*+1		
1006	01321	070517		LDA	A,I		GET MANTISSA
1007	01322	070210		SZA	*+4		CHECK IF MANTISSA WAS ZERO
1008	01323	020012		LDA	.1		
1009	01324	074111		SLB	*+2		
1010	01325	020132		LDA	M1		SKIP IF SIGN IS 0
1011	01326	070137		LDB	A		OTHERWISE SET A = -1
1012	01327	170402		RET			TRANSFER A INTO B
1013*							
1014	01330	031466	NPWR	STA	LOCLI		
1015	01331	021450		LDA	RMODE		
1016	01332	031702		STA	NPRIN		
1017	01333	021466		LDA	LOCLI		

143

PAGE 20

BASE PAGE ROUTINES

```
1018 01334 164167      JMP WRITE,I
1019*
1020 01335 035702 NPWRT STB NPRIN
1021 01336 164167      JMP WRITE,I
1022*
1023 01337 035701 NDWRT STB NDISP
1024 01340 164167      JMP WRITE,I
1025*
1026 01341 160170      JSM CLERA,I
1027 01342 021450 CRLF LDA PMODE
1028 01343 072450      RZA CRI
1029 01344 020031 CRLF1 LDA CRTN
1030 01345 061353      JSM TTY      OUTPUT CRTN
1031 01346 070742      SAR 16
1032 01347 160261      JSM IOSTA,I STOP KEY?
1033 01350 020040      LDA .31 YES, SET CRLF SUPRESSED FLAG
1034 01351 031622      STA TERM
1035 01352 020061      LDA LNFD OUTPUT LINE FEED
1036*
1037* GENERAL DRIVER
1038*
1039 01353 160261 TTY JSM IOSTA,I STOP KEY?
1040 01354 170402 CRI RET YES
1041 01355 041447 IOR SLCOD
1042 01356 172741 STF 1
1043 01357 172141 OTA 1
1044 01360 173601 STC 1,C
1045 01361 170402 RET
1046*
1047* CONVERT FROM LOWER CASE
1048*
1049 01362 061041 GTCON JSM GETCH
1050 01363 070137 LDB A
1051 01364 050103 AND B140
1052 01365 010103 CPA B140
1053 01366 004151 ADB M32
1054 01367 074117 LDA B
1055 01370 170402 RET
1056*
1057* TEMP1 IS POINTING AT ADDRESS OF THE VARIABLE IN SYMBOL TABLE
1058* ON RETURN (A) = TYPE CODE
1059* (B) = ADDRESS
1060*
1061 01371 025474 FIX LDB TEMP1
1062 01372 074517 LDA B,I
1063 01373 050040 AND B37
1064 01374 004132 ADB M1
1065 01375 074537 LDB B,I
1066 01376 170402 RET
1067*
1068*****
1069*
1070 01377 000000 BSS 1 CHECKSUM WORD FOR ADD.'S 1000 TO 1377
1071*
1072*****
1073*
```

PAGE 21

9830 — BASE PAGE READ-WRITE MEMORY

1075	01400		ORG	1400B	
1076	01400	000000	BADDR	BSS	1
1077	01401	000000	RSARE	BSS	1
1078	01402	000000	OTPTR	BSS	1
1079	01403	000000	MAW	BSS	1
1080	01404	000000	LWAM	BSS	1
1081	01405	000000	FWUP	BSS	1
1082	01406	000000	VALUE	BSS	1
1083	01407	000000	SYMTP	BSS	1
1084	01410	000000	LSTAK	BSS	1
1085	01411	000000	TSTPT	BSS	1
1086	01412	000000	LSTPT	BSS	1
1087	01413	000000	HSTPT	BSS	1
1088	01414	000000	DSTRT	BSS	1
1089	01415	000000	NXTDT	BSS	1
1090	01416	000000	MODE	BSS	1
1091	01417	000000	PARMA	BSS	1
1092	01420	000000	PARMN	BSS	1
1093	01421	000000	PRADD	BSS	1
1094	01422	000000	UKPTR	BSS	1
1095	01423	000000	BPTR	BSS	1
1096	01424	000000	PBUFF	BSS	1
1097	01425	000000	PBUFO	BSS	1
1098	01426	000000	PBPTO	BSS	1
1099	01427	000000	.LNUM	BSS	1
1100	01430	000000	LNINC	BSS	1
1101	01431	000000	CLINE	BSS	1
1102	01432	000000	DRQST	BSS	1
1103	01433	000000	TFLAG	BSS	1
1104	01434	000000	UNITS	BSS	1
1105	01435	000000	SEC	BSS	1
1106	01436	000000	HFLG1	BSS	2
1107	01440	000000	TFLG1	BSS	2
1108	01442	000000	STPFL	BSS	1
1109	01443	000000	KEYFG	BSS	1
1110	01444	000000	SAVEA	BSS	1
1111	01445	000000	SAVEB	BSS	1
1112	01446	000000	ADRSS	BSS	1
1113	01447	000000	SLCOD	BSS	1
1114	01450	000000	PMODE	BSS	1
1115	01451	000000	CTYPE	BSS	1
1116	01452	000000	CPLAC	BSS	1
1117	01453	000000	PRFLG	BSS	1
1118	01454	000000	RCOUT	BSS	1
1119	01455	000000	FORMT	BSS	1
1120	01456	000000	FORMS	BSS	1
1121	01457	000000	FORME	BSS	1
1122	01460	000000	FTMP1	BSS	1
1123	01461	000000	FTMP2	BSS	1
1124	01462	000000	REEDL	BSS	1
1125	01463	000000	INTEL	BSS	1
1126	01464	000000		BSS	2
1127*					
1128*					
1129*					
1130	01466	000000	LOCLI	BSS	1

CURRENT CHAR POINTER

SAVE (A) IN RETN2

REFRESH POINTER

LAST WORD OF R/W MEM

LAST WORD AVAILABLE FOR USER

FIRST WORD OF COMMON OR MAINLINE

FIRST WORD -1 OF VALUE TABLE

FIRST WORD -1 OF SYMBOL TABLE

FIRST WORD OF OPERATOR STACK

TEMPORARY STACK POINTER

OPERATOR STACK POINTER

OPERAND STACK POINTER

DATA STATEMENT

POINTERS

PROGRAM=0; KEYBOARD=1

PARAMETER ADDRESS

PARAMETER NAME

LAST WORD+1 OF CURRENT STMT

USER KEY POINTER

LAST WORD OF CURRENT PROGRAM

FIRST WORD OF CURRENT PROGRAM

FIRST WORD OF MAINLINE

LAST WORD+1 OF MAINLINE

CURRENT LINE NUMBER

AUTO LINE INCREMENT

CURRENT LINE FOR LIST

INPUT STMT FLAG

PTAPE FLAG

TRIG UNITS

SECURE PROGRAM FLAG

STOP ADDRESSES

TRACE LIMIT ADDRESSES

STOP FLAG

KEYBOARD FLAG

INTERRUPT

TEMPORARIES

SELECT CODE

PRINT-ALL FLAG

KEYBOARD

FORMAT FLAGS

PRINT/DISP FLAG

FORMAT

STATEMENT FLAGS

CURRENT OPERATOR CODE

CURRENT OPERATOR PRECEDENCE

KEYBOARD DRIVER LINK

INTERRUPT LINK

UNUSED

PAGE 22

9830 — BASE PAGE READ-WRITE MEMORY

1131	01467	000000	LOCL2	BSS	1	
1132	01470	000000	L2T1	BSS	1	
1133	01471	000000	L2T2	BSS	1	
1134	01472	000000	TEMPS	BSS	1	
1135	01473	000000	TEMP	BSS	1	
1136	01473		MBOX1	EQU	TEMP	
1137	01474	000000	TEMP1	BSS	1	
1138	01475	000000	TEMP2	BSS	1	
1139	01476	000000	TEMP3	BSS	1	
1140	01477	000000	TEMP4	BSS	1	
1141*						
1142*			BUFFERS			
1143*						
1144	01500	000000	IOBUF	BSS	41	INPUT AND RECALL BUFFER
1145	01551	000000	RBUF	BSS	41	SINGLE LINE REFRESH BUFFER
1146*						
1147*			PRINT ROUTINE FLAGS			
1148*						
1149	01622	000000	TERM	BSS	1	CRLF FLAG
1150	01623	000000	PEND	BSS	1	BUFFER OVERFLOW FLAG
1151	01624	000000	PPFF	BSS	1	BUFFER POINTER OR TAB FLAG
1152	01625	000000	FIRST	BSS	1	
1153	01626	000000		BSS	2	UNUSED
1154*						
1155*			MATH TEMPORARIES			
1156*						
1157	01630	000000	TREGE	BSS	4	
1158	01634	000000	RSLTE	BSS	4	
1159*						
1160*			STATEMENT SYNTAX AND EXECUTION TEMPORARIES			
1161*						
1162	01640	000000	ST	BSS	32	
1163*						
1164*			SYSTEM SYNTAX AND EXECUTION TEMPORARIES			
1165*						
1166	01700	000000		BSS	1	SAVED FOR PROCESSOR
1167	01701	000000	NDISP	BSS	1	
1168	01702	000000	NPRIN	BSS	1	
1169	01703	000000	PLACE	BSS	1	
1170	01704	000000	WIDTH	BSS	1	
1171	01705	000000	FLOAT	BSS	1	
1172	01706	000000	SBPTR	BSS	1	
1173	01707	000000	BLANK	BSS	1	
1174	01710	000000	MSFLG	BSS	1	
1175	01711	000000	DFLAG	BSS	1	
1176	01712	000000	PFLAG	BSS	1	
1177	01713	000000	UFLAG	BSS	1	
1178	01714	000000	SFLAG	BSS	1	
1179	01715	000000	SBADR	BSS	1	
1180	01716	000000	SSBPT	BSS	1	
1181	01717	000000	NUM	BSS	1	
1182	01717		OPTYP	EQU	NUM	
1183	01720	000000	LASTN	BSS	1	
1184	01721	000000	SIGN	BSS	1	
1185	01721		MAXIN	EQU	SIGN	
1186	01722	000000	EXP	BSS	1	

146

134

1,444,141

134

PAGE 23

9830 — BASE PAGE READ-WRITE MEMORY

1187	01722		INTGR	EQU	EXP
1188	01723	000000	ARYAD	BSS	1
1189	01724	000000	GFLAG	BSS	1
1190	01725	000000	NXTST	BSS	1
1191	01726	000000	MBIN1	BSS	1
1192	01727	000000	MBIN2	BSS	1

1193*

1194* ARITHMETIC REGISTERS AND RETURN STACK

1195*

1196	01740			ORG	1740B
1197	01740	000000	XC	BSS	4
1198	01744	000000	AR1	BSS	4
1199	01750	000000	YC	BSS	4
1200	01754	000000	AR2	BSS	4
1201	01760	000000	RSTAK	BSS	11
1202	01773	000000	RES	BSS	4
1203	01777	000000	RSPTR	BSS	1

RETURN STACK
RESULT REGISTER
RETURN STACK POINTER

PAGE 24

#01 EQUATES

1205	01466	MCAND	EQU	LOCL1
1206	01467	PROD	EQU	LOCL2
1207	01744	AR1E	EQU	AR1
1208	01754	AR2E	EQU	AR2
1209	01730	COML	EQU	MBIN2+1
1210	01731	TYPE	EQU	MBIN2+2
1211	01640	TEPPP	EQU	ST
1212	01641	SPRDD	EQU	ST+1
1213	01642	SAVE	EQU	ST+2
1214	01453	ARW	EQU	PRFLG
1215	01740	TMP	EQU	XC
1216	01741	TMP1	EQU	XC+1
1217	01742	TMP2	EQU	XC+2
1218	01743	TMP3	EQU	XC+3
1219	01750	TMP4	EQU	YC
1220	01751	TMP5	EQU	YC+1
1221	01706	TMP6	EQU	SBPTR
1222	01752	TMP6A	EQU	YC+2
1223	01460	TMP7	EQU	FTMP1
1224	01474	TMP9	EQU	TEMPS+2
1225	01475	TMP10	EQU	TEMPS+3
1226	01476	TMP11	EQU	TEMPS+4
1227	01477	TMP12	EQU	TEMPS+5
1228	01461	TMP13	EQU	FTMP2
1229	01473	CCNT	EQU	TEMPS+1
1230	01704	BKPTR	EQU	WIDTH
1231	01741	R	EQU	TMP1
1232	01742	L	EQU	TMP2
1233	01751	DIGIT	EQU	TMP5
1234	01743	INDEX	EQU	TMP3
1235	01750	RBASE	EQU	TMP4
1236*				
1237*				
1238	00047	PLUS	EQU	.43
1239	00050	MINUS	EQU	.45
1240	00051	POINT	EQU	.46
1241	00052	B60	EQU	.48
1242	00053	COLON	EQU	.58
1243	00077	DOLAR	EQU	B44
1244	00116	B2K	EQU	QTOP
1245	00004	XECA	EQU	B10K

(147)

PAGE 25

KEYBOARD ENTRY AND DISPLAY

```

1248*
1249* REED SUBROUTINE
1250*
1251* ENTRY CONDITIONS: NONE
1252*
1253* EXIT CONDITIONS:
1254*
1255* AN INPUT RECORD IS IN THE I/O
1256* BUFFER. THE LAST CHAR IS AN EOL.
1257*
1258*
1259* IF A DELETE LINE IS HIT
1260* RETURN TO P+1, OTHERWISE
1261* RETURN TO P+2.
1262*
1263*
1264 02000      ORG 2000B
1265*
1266*
1267 02000 020236 REED1 LDA SLEND
1268 02001 031623 STA PEND
1269 02002 020062 LDA FLGBT
1270 02003 031416 STA MODE SET TO PROGRAM SYNTAX
1271 02004 021433 LDA TFLAG
1272 02005 070110 SZA *+2 PTAPE COMMAND?
1273 02006 066617 JMP PREED
1274 02007 031473 STA CCNT
1275 02010 031704 STA BKPTR
1276* GET FIRST KEY
1277 02011 063046 REEDE JSM CALLK GET FIRST KEY
1278 02012 066011 JMP *-1
1279 02013 013127 CPA DELIN
1280 02014 066223 JMP DELI
1281 02015 025430 LDB LNINC
1282 02016 010074 CPA B22 BACKSPACE?
1283 02017 074112 SBM REE YES, SKIP IF LIST INPUT
1284 02020 066032 JMP REED7
1285 02021 055400 REE DSZ BADDR YES, BACKUP POINTER
1286 02022 045623 ISZ PEND
1287 02023 020041 LDA .32
1288 02024 061016 JSM ONEXT
1289 02025 055400 DSZ BADDR WIPE OUT LAZY-T
1290 02026 055623 DSZ PEND RESTORE END OF BUFFER POINTER
1291 02027 070742 SAR 16
1292 02030 031430 STA LNINC GET OUT OF AUTO
1293 02031 066045 JMP REED6
1294 02032 062533 REED7 JSM CLEAR CLEAR THE DISPLAY
1295 02033 021430 LDA LNINC
1296 02034 070153 SAP REED4 LIST WAS LAST?
1297 02035 070742 SAR 16
1298 02036 031430 STA LNINC GET OUT OF AUTO
1299 02037 070210 REED4 SZA REED6-2 AUTO LINE?
1300 02040 025427 LDB LNUM YES OUTPUT #
1301 02041 160210 JSM OUTIA,I
1302 02042 045400 ISZ BADDR OUTPUT A BLANK
1303 02043 021751 LDA TMP5

```


148

136

1,444,141

136

PAGE 26

KEYBOARD ENTRY AND DISPLAY

1304	02044	066047		JMP REEDG	
1305*	GET NEXT KEY				
1306	02045	063046	REED6	JSM CALLK	GET NEXT KEY
1307	02046	066045		JMP *-1	
1308	02047	010074	REEDG	CPA B22	BACKSPACE?
1309	02050	066707		JMP BKSP	
1310	02051	010075		CPA B23	FORWARD?
1311	02052	066725		JMP FORWD	YES
1312	02053	012436		CPA B152	CLEAR?
1313	02054	067665		JMP ECLR	YES
1314	02055	010027		CPA .11	EXECUTE KEY?
1315	02056	045416		ISZ MODE	YES
1316	02057	050103		AND MSK65	
1317	02060	072510		RZA REED2	BITS 5&6=0?
1318	02061	021751		LDA TMP5	YES
1319	02062	000141		ADA M10	
1320	02063	070413		SAP REED2+1	DEF KEY?
1321	02064	025706		LDB TMP6	
1322	02065	074210		SZB *+4	
1323	02066	021751		LDA TMP5	
1324	02067	040101		IOR B100	SET SHIFT BIT FOR DEF-KEYS
1325	02070	031751		STA TMP5	
1326	02071	067131		JMP KEYPR	
1327	02072	010103	REED2	CPA MSK65	BITS 5&6=1?
1328	02073	066103		JMP REED3	YES
1329	02074	021706		LDA TMP6	NO
1330	02075	070610		SZA REED5	SHIFTED?
1331	02076	021751		LDA TMP5	
1332	02077	060752		JSM LETCK	YES, LOWER CASE LETTER?
1333	02100	066103		JMP REED3	NO
1334	02101	040041		IOR .32	CONVERT CODE
1335	02102	066110		JMP REED5-1	
1336	02103	026412	REED3	LDB VALUP	
1337	02104	063071		JSM TBSR1	CODE IN VALUE TABLE?
1338	02105	066124		JMP REEDD	NO
1339	02106	010022		CPA .5	ENTER EXPONENT?
1340	02107	000101		ADA B100	YES, FORM ASCII E
1341	02110	031751		STA TMP5	
1342*					
1343*	ALL CONVERSIONS COMPLETE				
1344	02111	021751	REED5	LDA TMP5	
1345	02112	010031		CPA CRTN	
1346	02113	066172		JMP EOLL	EXECUTE OR STORE
1347	02114	025704		LDB BKPTR	
1348	02115	074210		SZB REED9	CURSOR?
1349	02116	027122		LDB BKPTA	YES
1350	02117	061017		JSM OUTCR	REPLACE CHARACTER
1351	02120	066730		JMP FORWD+3	SCROLL IF NECESSARY
1352	02121	061016	REED9	JSM ONEXT	PUT CHAR AT END OF LINE
1353	02122	062737		JSM LEFT	SCROLL IF NECESSARY
1354	02123	066045		JMP REED6	
1355*					
1356	02124	026373	REEDD	LDB EXPTB	SEARCH FOR MULTI-CHAR KEY
1357	02125	063071		JSM TBSR1	FOUND?
1358	02126	066213		JMP REED8	NO
1359	02127	031751		STA TMP5	

PAGE 27 KEYBOARD AND DISPLAY

1360	02130	062521	JSM	CLRST	CLEAR FLAG IF "STOP"	
1361	02131	062533	JSM	CLEAR		
1362	02132	026443	LDB	COMTA		
1363	02133	021751	LDA	TMP5		
1364	02134	162441	JSM	MOUTA,I	OUTPUT FROM SYSTEM COMMAND TABLE	
1365	02135	066045	JMP	REED6		
1366	02136	062533	PRALL JSM	CLEAR	CLEAR BUFFER	
1367	02137	021450	LDA	PMODE		
1368	02140	070130	SIA	*+2		
1369	02141	070742	SAR	16		
1370	02142	031450	STA	PMODE		
1371	02143	026444	LDB	OFFA	GET ADDRESS OF "OFF"	
1372	02144	072110	RZA	*+2	ON?	
1373	02145	004020	ADB	.3	YES, GET ADDRESS OF "ON"	
1374	02146	020020	LDA	.3		
1375	02147	061335	JSM	NPWRT		
1376	02150	066011	JMP	REED6	GET FIRST KEY AGAIN	
1377*						
1378*	STEP AND CONT EXECUTION					
1379*						
1380	02151	031442	STEP	STA	STPFL	SET POSITIVE NON-ZERO
1381	02152	070742	SAR	16		
1382	02153	025421	CONT	LDB	PRADD	
1383	02154	074273	SBP	CONT1,C	HAS AN EDIT OCCURRED?	
1384	02155	072110	RZA	CONTK	YES, LINE # GIVEN?	
1385	02156	074517	LDA	B,I	NO, EXECUTE FROM BEGINNING	
1386	02157	031725	CONTK	STA	NXTST	YES, EXECUTE FROM BEGINNING
1387	02160	164004	JMP	XECA,I		
1388	02161	070250	CONT1	SZA	*+5	LINE # GIVEN?
1389	02162	061075	JSM	FNDPS	YES, FIND ADDRESS	
1390	02163	164174	JMP	RDYDA,I		
1391	02164	066165	JMP	*+1		
1392	02165	035421	STB	PRADD	STORE ADDRESS	
1393	02166	062533	JSM	CLEAR		
1394	02167	031416	STA	MODE	SET TO PROGRAM EXECUTION	
1395	02170	025421	LDB	PRADD		
1396	02171	164255	JMP	XEC6A,I	CONTINUE EXECUTION	
1397*						
1398*	PROCESS EXEC OR STORE					
1399	02172	045623	EOLL	ISZ	PEND	GUARANTEE ROOM FOR EOL
1400	02173	061016	JSM	ONEXT		
1401	02174	024233	LDB	BEND		
1402	02175	035477	STB	TEMP4		
1403	02176	024233	LDB	SBUFA		
1404	02177	035476	STB	TEMP3		
1405	02200	004046	ADB	.41		
1406	02201	061122	JSM	MVTOH+1	MOVE INPUT RECORD TO I/O BUFF	
1407	02202	060373	JSM	ICNT		
1408	02203	000132	ADA	M1		
1409	02204	070250	SZA	REEDT	SKIP IF NULL RECORD	
1410	02205	025450	LDB	PMODE		
1411	02206	076150	RZB	REEDT		
1412	02207	024211	LDB	BUFA		
1413	02210	061337	JSM	NDWRT		
1414	02211	062502	REEDT	JSM	CRC	
1415	02212	064357	JMP	RETN2		

150

138

1,444,141

138

PAGE 28

KEYBOARD ENTRY AND DISPLAY

```

1416*
1417 02213 013124 REED8 CPA B33 DIVIDE?
1418 02214 022437 LDA B57 YES, GET ASCII
1419 02215 026361 LDB EDIT2
1420 02216 063072 JSM TBSRH CODE IN EDIT TABLE?
1421 02217 066110 JMP REED5-1 NO
1422 02220 002440 ADA BEEB
1423 02221 070737 JMP A,I
1424*
1425* DELETE LINE
1426 02222 170402 DEL2 RET
1427 02223 021430 DEL1 LDA LNINC
1428 02224 071713 SAP DEL2
1429 02225 021431 LDA CLINE
1430 02226 130233 STA SBUFA,I
1431 02227 070742 SAR 16
1432 02230 031430 STA LNINC
1433 02231 164204 JMP DELSA,I
1434*
1435* RECALL
1436*
1437 02232 062533 EREC JSM CLEAR
1438 02233 055473 DSZ CCNT SET RECALL FLAG
1439 02234 024211 LDB BUFA
1440 02235 035740 RSLT1 STB TMP GET STARTING ADDRESS OF IO BUFF
1441 02236 066045 JMP REED6
1442*
1443* INITIALIZE VARIABLES
1444*
1445 02237 020056 INITV LDA IE4
1446 02240 164171 JMP RUNA,I RUN 10000
1447* RESULT KEY
1448 02241 020101 RSLT LDA B100 CODE FOR
1449 02242 025706 LDB TMP6
1450 02243 074110 SZB *+2 SHIFTED?
1451 02244 066110 JMP REED5-1 YES
1452 02245 026445 LDB RESA
1453 02246 020020 RFNX1 LDA 3
1454 02247 031473 STA CCNT COUNT=3
1455 02250 066235 JMP RSLT1
1456*
1457*
1458 02251 066136 PAL JMP PRALL
1459 02252 066151 STPL JMP STEP

```

151

PAGE 29

KEYBOARD ENTRY AND DISPLAY

1461*
 1462* **FETCH EXECUTION**
 1463*
 1464 02253 000132 EFET ADA M1
 1465 02254 031431 STA CLINE
 1466*
 1467* **THIS ROUTINE LISTS ONE LINE**
 1468* **OF THE PROGRAM. THE FIRST**
 1469* **LINE TO BE LISTED IS THE FIRST**
 1470* **LINE OF THE PROGRAM.**
 1471*
 1472*
 1473 02255 062353 DNARW JSM SAROW **CHECK FOR NULL PROGRAM**
 1474 02256 074310 SZB U4 **SKIP IF TYPING AID KEY**
 1475 02257 070070 SIA *+1
 1476 02260 061075 JSM FNDPS **FIND LINE**
 1477 02261 066277 JMP U3 **PAST END OF PROGRAM**
 1478 02262 066301 JMP U5-1 **BETWEEN LINES**
 1479 02263 066301 JMP U5-1
 1480*
 1481* **THIS ROUTINE LISTS ONE LINE**
 1482* **OF THE PROGRAM. THE FIRST LINE**
 1483* **TO BE LISTED IS THE LAST LINE**
 1484* **OF THE PROGRAM.**
 1485*
 1486*
 1487 02264 021424 U4 LDA PBUFF **GET FIRST LINE.**
 1488 02265 066302 JMP U5
 1489 02266 062353 UPARW JSM SAROW **CHECK FOR NULL PROG**
 1490 02267 075650 SZB U4 **SKIP IF TYPING AID**
 1491 02270 072110 RZA U1 **CLINE = 0?**
 1492 02271 020056 LDA 1E4 **YES, FIND LAST LINE OF PROG**
 1493 02272 061075 JSM FNDPS **FIND LINE**
 1494 02273 066277 JMP U3 **PAST END OF PROG**
 1495 02274 066275 JMP *+1 **BETWEEN LINES**
 1496 02275 015424 CPB PBUFF **FIRST LINE?**
 1497 02276 066301 JMP U5-1 **YES**
 1498 02277 021720 U3 LDA LASTN **GET PREVIOUS LINE**
 1499 02300 060742 JSM FND
 1500 02301 074117 LDA B **GET ADDRESS IN (A)**
 1501 02302 074742 U5 SBR 16
 1502 02303 035453 STB ARW **FLAG LIST ROUTINE**
 1503 02304 035460 STB TMP7
 1504 02305 167120 JMP LISTR,I
 1505*
 1506* **INSERT AND DELETE CHARACTER**
 1507*
 1508 02306 066045 JMP REED6
 1509 02307 025704 INSRT LDB BKPTR
 1510 02310 075710 SZB INSRT-1 **ANY BACKSPACE YET?**
 1511 02311 025400 LDB BADDR
 1512 02312 021706 LDA TMP6
 1513 02313 070710 SZA INS0 **SHIFT FLAG SET?**
 1514* **DELETE**
 1515 02314 055400 DSZ BADDR **YES, DINC END OF LINE POINTER**
 1516 02315 025704 LDB BKPTR

PAGE 30

KEYBOARD ENTRY AND DISPLAY

1517	02316	035740		STB	TMP	SET POINTER TO CURSOR CHARACTER
1518	02317	025740	DELC1	LDB	TMP	
1519	02320	015400		CPB	BADDR	LAST CHAR?
1520	02321	066347		JMP	INS2+1	YES, OUTPUT A BLANK
1521	02322	045740		ISZ	TMP	
1522	02323	061053		JSM	GTMP	GET CHARACTER
1523	02324	055740		DSZ	TMP	
1524	02325	055740		DSZ	TMP	
1525	02326	024111		LDB	TMPA	STORE SHIFTED LEFT ONE
1526	02327	061017		JSM	OUTCR	
1527	02330	066317		JMP	DELC1	
1528*	INSERT					
1529	02331	014236	INS0	CPB	SLEND	
1530	02332	066045		JMP	REED6	LINE FULL
1531	02333	004132		ADB	M1	
1532	02334	035740	INS1	STB	TMP	
1533	02335	061053		JSM	GTMP	GET CHARACTER
1534	02336	025740		LDB	TMP	
1535	02337	015704		CPB	BKPTR	AT CURSOR POSITION?
1536	02340	066346		JMP	INS2	YES
1537	02341	024111		LDB	TMPA	NO, MOVE RIGHT 1 PLACE
1538	02342	061017		JSM	OUTCR	
1539	02343	025740		LDB	TMP	
1540	02344	004134		ADB	M3	BACK UP POINTER TO NEXT
1541	02345	066334		JMP	INS1	CHAR LEFT
1542	02346	045400	INS2	ISZ	BADDR	UPDATE END OF LINE POINTER
1543	02347	020041		LDA	.32	
1544	02350	024111		LDB	TMPA	
1545	02351	061017		JSM	OUTCR	INSERT A BLANK
1546	02352	066045		JMP	REED6	
1547*						
1548*	SUBROUTINE FOR UP AND DOWN ARROW ROUTINES					
1549*						
1550	02353	021424	SAROW	LDA	PBUFF	
1551	02354	011423		CPA	PBPTR	
1552	02355	164174		JMP	RDYDA,I	NULL PROGRAM ABORT
1553	02356	021431		LDA	CLINE	
1554	02357	125424		LDB	PBUFF,I	
1555	02360	170402		RET		

PAGE 31

KEYBOARD ENTRY AND DISPLAY

1557*

1558*

1559*

1560*

1561*

1562*

1563*

1564

1565

1566

1567

1568

1569

1570

1571

1572

1573

1574

1575*

1576*

1577*

1578*

1579*

1580

1581

1582

1583

1584

1585

1586

1587

1588

1589

1590

1591

1592

1593

1594

1595*

1596*

1597*

1598*

1599

1600

1601

1602

1603

1604

1605

1606

1607

1608

1609

1610

1611

1612

THIS SECTION CONTAINS TABLES,
POINTERS USED IN THE ABOVE ROUTINE

02361	004744	EDIT2	DEF	*+*+2
02362	012643		DEF	12600B-EEB+UPARW
02363	012232		DEF	12200B-EEB+DNARW
02364	015214		DEF	15200B-EEB+INITV
02365	065616		DEF	65600B-EEB+RSLT
02366	006627		DEF	6600B-EEB+STPL
02367	017664		DEF	17600B-EEB+INSRT
02370	071600		DEF	71600B-EEB+DEL1
02371	070226		DEF	70200B-EEB+PAL
02372	010307		DEF	10300B-EEB+EREC
02223		EEB	EQU	DEL1

EXPANDING DEY TABLE

02373	004770	EXPTB	DEF	*+*+2	
02374	007601	OCT	7601		RUN
02375	014600	OCT	14600		STORE
02376	014220	OCT	14220		LOAD
02377	017207	OCT	17207		NORMAL
02400	006212	OCT	6212		CONTINUE
02401	007226	OCT	7226		TRACE
02402	077637	OCT	77637		STOP
02403	017216	OCT	17216		NORMAL
02404	010604	OCT	10604		FETCH
02405	072214	OCT	72214		FIXED
02406	072615	OCT	72615		FLOAT
02407	005203	OCT	5203		LIST
02410	073202	OCT	73202		SCRATCH
02411	073705	OCT	73705		AUTO

KEY VALUE TABLE

02412	005026	VALUP	DEF	*+*+2	
02413	005615	OCT	5615		EOL
02414	071215	OCT	71215		STORE
02415	020240	OCT	20240		BLANK
02416	030641	OCT	30641		!
02417	032244	OCT	32244		\$
02420	032645	OCT	32645		%
02421	067605	OCT	67605		E
02422	031643	OCT	31643		#
02423	031242	OCT	31242		"
02424	033246	OCT	33246		&
02425	033647	OCT	33647		'
02426	034250	OCT	34250		(
02427	034651	OCT	34651		)

PAGE 32

KEYBOARD ENTRY AND DISPLAY

1613	02430	027677	OCT	27677	?	
1614	02431	026274		OCT	26274	<
1615	02432	027276		OCT	27276	>
1616	02433	035252	OCT	35252	*	
1617	02434	035653	OCT	35653	+	
1618	02435	030355	OCT	30355	-	
1619*						
1620*						
1621*						
1622	00103		MSK65	EQU	B140	
1623	02436	000152	B152	OCT	152	
1624	02437	000057	B57	OCT	57	
1625	02440	002223	BEEB	DEF	EEB	
1626	02441	006306	MOUTA	DEF	MCOUT	
1627	02442	003402	LPTR	DEF	RBUFF+RBUFF+48	
1628	02443	007062	COMTA	DEF	LBL1+LBL1+20	
1629	02444	005117	OFFA	DEF	*+*+7	
1630	02445	005114	RESA	DEF	*+*+2	
1631	02446	051105		ASC	4,RESOFFON	
	02447	051517				
	02450	043106				
	02451	047516				
1632	02452	020040	TOBLN	OCT	20040	
1633*						
1634*						
1635	02453	000000		BSS	1	*****CHECKSUM*****

155

PAGE 33

KEYBOARD ENTRY AND DISPLAY

1637* WRITE SUBROUTINE

1638*

1639* ENTRY CONDITIONS:

1640*

1641*

1642*

1643*

1644*

1645*

1646*

1647

1648

1649

1650

1651

1652

1653

1654

1655

1656

1657

1658

1659

1660

1661

1662

1663

1664

1665

1666

1667

1668

1669*

1670

1671

1672

1673

1674

1675

1676

1677

1678

1679

1680

1681

1682*

1683

1684

1685

1686*

1687

1688

1689

1690

1961

1692

(A)=CHARACTER COUNT
 (B)=POINTER TO FIRST CHAR
 NPRIN#0 FOR PRINT ABORT
 NDISP#0 PRINT ONLY

WRIT

RITE1

RIT3L

RITE4

RITE2

RITE3

RETA

CRC

CRCHK

CRC2

CRC1

CTERM

CLRST

SZA

STA

STB

JSM

CPA

LDA

LDB

CPA

JMP

RZB

SZA

JSM

LDA

LDB

RZB

JSM

DSZ

JMP

SAR

STA

STA

RET

LDB

SZB

LDB

SZB

LDB

SBM

LDB

SZB

SFC

JMP

JSM

JSM

SBR

STB

RET

LDB

CPB

JMP

RET

SBR

STB

RIT3L

TMP1

TMP

GTMP

.11

B52

NDISP

CRTN

RITE3

RITE2-1

RITE4

ONEXT

LOCL2

NPRIN

RITE2

TTY

TMP1

RITE1

16

NPRIN

NDISP

PMODE

CRC1

PMODE

CRC2

PRFLG

CTERM

TERM

RETA

0

*-1

CLRST

CRLF1

16

TERM

STPFL

.31

*+2

16

STPFL

SKIP IF NULL COUNT

INITIALIZE POINTER

GET CHARACTER

EXECUTE?

YES, CHANGE TO * FOR KEY

CRTN CHARACTER?

YES, ABORT

DISPLAY DISABLED?

NO, NULL?

OUTPUT TO DISPLAY BUFFER

RECALL CHARACTER

PRINT DISABLED?

NO, PRINT CHAR

MORE CHARACTERS?

YES

RESET FLAGS

PRINT ALL?

PRINT ALL?

WAIT UNTIL "STOP" KEY

IS RELEASED

CLEAR "STOP" FLAG

SET FROM KEYBOARD?

YES, CLEAR IT

PAGE 34

KEYBOARD ENTRY AND DISPLAY

```

1693 02527 035433      STB TFLAG
1694 02530 170402      RET
1695*
1696*
1697 02531 027117 GTMP2 LDB TMP2A
1698 02532 065041      JMP  GETCH
1699*
1700*
1701* THIS SUBROUTINE CLEARS THE DISPLAY
1702*
1703 02533 070742 CLEAR SAR 16
1704 02534 031704      STA BKPTR
1705 02535 020212      LDA SLBF
1706 02536 031402      STA OTPTR
1707 02537 031400      STA BADDR
1708 02540 031625      STA FIRST
1709 02541 024233      LDB BEND
1710 02542 022452      LDA TOBLN
1711 02543 074557 CLEI STA B,I
1712 02544 017123      CPB ENDW
1713 02545 066476      JMP RITE3
1714 02546 077670      RIB CLEI NO
1715*
1716* I/O STATUS CHECK ROUTINE
1717*
1718 02547 025442 IOST LDB STPFL
1719 02550 014040      CPB 31
1720 02551 170402      RET STOP FLAG?
1721 02552 025447      LDB SLCOD YES
1722 02553 172741      STF 1 NO
1723 02554 176141      OTB 1 TURN INTERRUPT OFF
1724 02555 172741      STF 1
1725 02556 177241      LIB 1,C GET STATUS IN I/O REG
1726 02557 074406      RBR 9 TURN INTERRUPT ON
1727 02560 075353      SBP IOST
1728 02561 064357      JMP RETN2 DEVICE BUSY?

```

PAGE 35 KEYBOARD ENTRY AND DISPLAY

```

1730*
1731*
1732*   THIS SUBROUTINE REFRESHES
1733*   THE SINGLE LINE DISPLAY
1734*
1735   02562  021402  REFRS  LDA  OTPTR
1736   02563  031740          STA  TMP
1737   02564  000033          ADA  .16
1738   02565  031742          STA  TMP2
1739   02566  074742          SBR  16
1740   02567  035741  RE1    STB  TMP1
1741   02570  024111          LDB  TMPA
1742   02571  061362          JSM  GTCON
1743   02572  070304  RE3    SAL  10
1744   02573  031743          STA  TMP3
1745   02574  027117          LDB  TMP2A
1746   02575  061362          JSM  GTCON
1747   02576  050054  RE4    AND  .63
1748   02577  070604          SAL  4
1749   02600  041743          IOR  TMP3
1750   02601  041741          IOR  TMP1
1751   02602  173750          CLF  8
1752   02603  173741          CLF  1
1753   02604  025443          LDB  KEYFG
1754   02605  172741          STF  1
1755   02606  074353          SBP  RE2
1756   02607  025741          LDB  TMP1
1757   02610  014033          CPB  .16
1758   02611  066615          JMP  RE2
1759   02612  074070          SIB  *+1
1760   02613  172150          OTA  8
1761   02614  066567          JMP  RE1
1762   02615  173741  RE2    CLF  1
1763   02616  170402          RET

FORM OUTPUT WORD
TURN DISPLAY OFF
INTERRUPT ON
INTERRUPT OFF
INTERRUPT OCCURRED?
TURN INTERRUPT ON

```

PAGE 36

KEYBOARD ENTRY AND DISPLAY

```

1765*
1766*   PTAPE COMMAND EXECUTION
1767*
1768*   THIS ROUTINE READS PROGRAMS FROM
1769*   THE PHOTOREADER. 10 OR MORE NULLS
1770*   CONSTITUTE LEADER OR TRAILER WHICH
1771*   ARE THE PROGRAM DELIMITERS.
1772*
1773*   INPUT CODES:
1774*
1775*       DATA 64 ASCII SET
1776*
1777*       CONTROL
1778*
1779*           176 ALT MODE=DELETE LINE
1780*           33  ESC MODE=DELETE LINE
1781*           137 LEFT ARROW=DELETE CHAR
1782*           177 RUBOUT=IGNORE
1783*           15  CRTN=IGNORE
1784*           12  LNFD=END OF RECORD
1785*   RETURN TO:
1786*
1787*       P+1 IF NULL RECORD
1788*       P+2 OTHERWISE
1789   02617 070204 PREED SAL 12
1790   02620 031447 STA SLCOD
1791   02621 020211 LDA BUFA
1792   02622 031400 STA BADDR
1793   02623 020132 LDA M1
1794   02624 031751 STA TMP5
1795   02625 020141 LDA M10
1796   02626 031741 STA TMP1
1797*
1798*
1799*   GENERAL INPUT ROUTINE
1800*
1801   02627 062547 P1 JSM IOST STOP KEY?
1802   02630 066704 JMP P7 YES
1803   02631 020121 LDA B4000 GET INPUT STATUS BIT
1804   02632 061355 JSM CR1+1 CALL FOR INPUT
1805   02633 062547 JSM IOST
1806   02634 066704 P4 JMP P7 STOP
1807   02635 074117 LDA B
1808   02636 070306 RAR 7
1809   02637 050104 AND B177
1810   02640 072310 RZA P3
1811   02641 045741 ISZ TMP1 NULL?
1812   02642 066627 JMP P1 10 FEED HOLES?
1813   02643 021742 LDA TMP2 NO
1814   02644 071413 SAP P4
1815   02645 066625 JMP P1-2 TRAILER?
1816   02646 031742 P3 STA TMP2 NO, LEADER
1817   02647 010104 CPA B177 RESET LEADER FLAG
1818   02650 066625 JMP P1-2 RUBOUT?
1819   02651 010031 CPA CRTN YES
1820   02652 066625 JMP P1-2 YES, IGNORE

```

159

PAGE 37

KEYBOARD ENTRY AND DISPLAY

1821	02653	013126	CPA	B137	LEFT ARROW?
1822	02654	066700	JMP	P5	YES, BACKSPACE
1823	02655	010061	CPA	LNFD	LINE FEED?
1824	02656	020031	LDA	EOL	YES, CHANGE TO END OF LINE CHARACTER
1825	02657	013124	CPA	B33	ESCAPE MODE?
1826	02660	066624	JMP	P1-3	YES, SET DELETE FLAG
1827	02661	013125	CPA	B176	ALT MODE?
1828	02662	066624	JMP	P1-3	YES, DELETE
1829	02663	061016	JSM	ONEXT	
1830	02664	021467	LDA	LOCL2	
1831	02665	010031	CPA	EOL	END OF RECORD?
1832	02666	066673	JMP	P2	YES
1833	02667	025400	LDB	BADDR	
1834	02670	017130	CPB	IOEND	FULL?
1835	02671	055400	DSZ	BADDR	
1836	02672	066625	JMP	P1-2	
1837	02673	020127	LDA	B170K	
1838	02674	031447	STA	SLCOD	SET TO STANDARD PRINTER
1839	02675	045751	ISZ	TMP5	CHECK DELETE FLAG
1840	02676	170402	RET		DELETE
1841	02677	064357	JMP	RETN2	ACCEPT
1842	02700	025400	LDB	BADDR	
1843	02701	014211	CPB	BUFA	BUFFER EMPTY?
1844	02702	066625	JMP	P1-2	YES, IGNORE
1845	02703	066671	JMP	P6	NO, BACKSPACE
1846	02704	020031	LDA	EOL	
1847	02705	061016	JSM	ONEXT	PUT EOL CHAR IN BUFFER
1848	02706	164174	JMP	RDYDA,1	EXIT PTAPE MODE

PAGE 38

KEYBOARD ENTRY AND DISPLAY

```

1850*
1851*   BACKSPACE
1852*
1853   02707  021704  BKSP   LDA  BKPTR
1854   02710  072350          RZA  BK2    FIRST BACKSPACE?
1855   02711  021400          LDA  BADDR   YES
1856   02712  010212          CPA  SLBF    NULL RECORD?
1857   02713  066045          JMP  REED6   YES
1858   02714  021400          LDA  BADDR
1859   02715  031704          STA  BKPTR
1860   02716  066723          JMP  BK1-1  PUT POINTER AT LAST CHAR
1861   02717  010212  BK2    CPA  SLBF    ALREADY AT FIRST CHAR?
1862   02720  066045          JMP  REED6   YES
1863   02721  011402          CPA  OTPTR   LEFTMOST ON SCREEN?
1864   02722  055402          DSZ  OTPTR   YES, SCROLL
1865   02723  055704          DSZ  BKPTR   BACKSPACE CURSOR
1866   02724  066045  BK1    JMP  REED6
1867*
1868*   FORWARD SPACE
1869*
1870   02725  021704  FORWD  LDA  BKPTR
1871   02726  070310          SZA  FW1    SKIP IF NO BACKSPACE HAS OCCURRED
1872   02727  045704          ISZ  BKPTR
1873   02730  021402          LDA  OTPTR
1874   02731  000041          ADA  32
1875   02732  011704          CPA  BKPTR   ON RIGHT OF SCREEN?
1876   02733  011400          CPA  BADDR   YES, LINE FULL?
1877   02734  066045  FW1    JMP  REED6
1878   02735  045402          ISZ  OTPTR   NO, SCROLL
1879   02736  066045          JMP  REED6
1880*
1881*   LEFT SUBROUTINE
1882*
1883*   SHIFT THE 32 CHARACTER DISPLAY LEFT
1884*
1885   02737  021400  LEFT    LDA  BADDR  IS THE LAST CHAR
1886   02740  000151          ADA  M32    ALREADY TO THE LEFT OF
1887   02741  070076          TCA
1888   02742  001402          ADA  OTPTR   THE END OF DISPLAY?
1889   02743  070113          SAP  *+2
1890   02744  045402          ISZ  OTPTR   NO, SCROLL
1891   02745  170402          RET
1892*
1893*   RIGHT SUBROUTINE
1894*
1895*   SHIFT THE 32 CHARACTER DISPLAY RIGHT
1896*   ONE POSITION. IF THE FIRST CHARACTER
1897*   IS IN THE LEFTMOST POSITION, ABORT.
1898*
1899   02746  021402  RIGHT    LDA  OTPTR
1900   02747  010212          CPA  SLBF
1901   02750  170402          RET
1902   02751  055402          DSZ  OTPTR   NO
1903   02752  170402          RET

```

PAGE 39

KEYBOARD ENTRY AND DISPLAY

1905* KEYIN SUBROUTINE

1906*
 1907* THIS SUBROUTINE SCANS ALL
 1908* INPUT DEVICES SEARCHING FOR
 1909* AN INPUT CODE. IT REFRESHES
 1910* THE DISPLAY WHILE WAITING
 1911* EXIT WITH CODE IN (A)
 1912*

1913	02753	074742	KEYIN	SBR	16	
1914	02754	021400		LDA	BADDR	
1915	02755	011704		CPA	BKPTR	CURSOR PAST END?
1916	02756	035704		STB	BKPTR	YES, CLEAR POINTER
1917	02757	021473	KEYN7	LDA	CCNT	
1918	02760	072110		RZA	*+2	
1919	02761	067002		JMP	KEYN1	
1920	02762	061053		JSM	GTMP	GET CHARACTER
1921	02763	031751		STA	TMP5	SAVE UNCONVERTED CHAR
1922	02764	070550		SZA	KEYN1-3	EMPTY BUFFER?
1923	02765	061363		JSM	GTCON+1	CONVERT
1924	02766	011751		CPA	TMP5	DID CONVERSION OCCUR?
1925	02767	066771		JMP	*+2	NO
1926	02770	043121		IOR	B200	YES, SET SHIFT BIT
1927	02771	025473		LDB	CCNT	
1928	02772	010031		CPA	EOL	END OF LINE?
1929	02773	074212		SBM	*+4	RECALL MODE?
1930	02774	055473		DSZ	CCNT	NO, DINC POINTER
1931	02775	170402		RET		
1932	02776	170402		RET		RETURN WITH CHAR IN (A)
1933	02777	074742		SBR	16	
1934	03000	035473		STB	CCNT	RESET POINTER
1935	03001	066756		JMP	KEYN7-1	
1936	03002	031750	KEYN1	STA	TMP4	SET CURSOR COUNTER TO ZERO
1937	03003	063024		JSM	KEYN6	
1938	03004	025704		LDB	BKPTR	
1939	03005	074250		SZB	KEYN3	CURSOR?
1940	03006	027122		LDB	BKPTA	YES
1941	03007	061041		JSM	GETCH	
1942	03010	031751		STA	TMP5	SAVE CHAR AT CURSOR POSITION
1943	03011	055704		DSZ	BKPTR	
1944	03012	025443	KEYN3	LDB	KEYFG	
1945	03013	074612		SBM	KEYN2	HAS KEY INPUT OCCURRED?
1946	03014	021704		LDA	BKPTR	YES
1947	03015	070250		SZA	KEYN5	CURSOR?
1948	03016	021751		LDA	TMP5	YES, RESTORE CHARACTER
1949	03017	027122		LDB	BKPTA	
1950	03020	061017		JSM	OUTCR	
1951	03021	055704		DSZ	BKPTR	
1952	03022	021443	KEYN5	LDA	KEYFG	
1953	03023	050105		AND	B377	
1954	03024	024132	KEYN6	LDB	M1	
1955	03025	035443		STB	KEYFG	RESET KEY OCCURRED FLAG
1956	03026	170402		RET		
1957	03027	025704	KEYN2	LDB	BKPTR	
1958	03030	074610		SZB	KEYN4	CURSOR?
1959	03031	025750		LDB	TMP4	YES
1960	03032	004132		ADB	M1	

KEYBOARD ENTRY AND DISPLAY

1961	03033	014136	CPB	M5	
1962	03034	024061	LDB	10	
1963	03035	035750	STB	TMP4	
1964	03036	021751	LDA	TMP5	
1965	03037	074113	SBP	*+2	
1966	03040	020037	LDA	28	
1967	03041	027122	LDB	BKPTA	
1968	03042	061017	JSM	OUTCR	REPLACE CHARACTER WITH CURSOR
1969	03043	055704	DSZ	BKPTR	
1970	03044	062562	JSM	REFRS	
1971	03045	067012	JMP	KEYN3	
1972*					
1973*	CALL KEYIN AND EXECUTE RIGHT AND LEFT ARROWS				
1974*					
1975*	P+1=ARROW; P+2=OTHERWISE				
1976	03046	062753	CALLK	JSM	KEYIN
1977	03047	074742	SBR	16	
1978	03050	070346	RAR	8	
1979	03051	070133	SAP	*+2,C	SHIFTED?
1980	03052	024132	LDB	M1	YES
1981	03053	035706	STB	TMP6	
1982	03054	070346	RAR	8	
1983	03055	031751	STA	TMP5	
1984	03056	010036	CPA	B27	RIGHT ARROW
1985	03057	066746	JMP	RIGHT	YES
1986	03060	010035	CPA	B26	
1987	03061	066737	JMP	LEFT	LEFT ARROW
1988	03062	064357	JMP	RETN2	

PAGE 41

KEYBOARD ENTRY AND DISPLAY

1990*

1991*

1992*

1993*

1994*

1995*

1996*

1997*

1998*

1999*

TABLE SEARCH ROUTINE

THIS SUBROUTINE COMPARES THE
INPUT STRING WITH STRINGS IN
A TABLE. IF A SUCCESSFUL COMPARE
IS MADE AN 6-BIT CODE IS RETURNED

ENTRY CONDITIONS:

(A)=FIRST CHARACTER OF INPUT STRING
(B)=POINTER TO FIRST CHAR OF TABLE

EXIT CONDITIONS:

IF MATCH, RETURN TO P+2 WITH CODE
IN A OTHERWISE RETURN TO P+1

2000*

2001*

2002*

2003*

2004*

2005*

2006*

2007*

2008

2009

2010

2011

2012

2013

2014

2015

2016

2017

2018

2019

2020

2021

2022

2023

2024

2025

2026

2027

2028

2029

2030

2031

2032

2033

2034

2035

03063 062531

03064 070306

03065 070511

03066 070402

03067 050054

03070 064357

03071 021751

03072 031750

03073 035742

03074 025400

03075 035752

03076 067101

03077 055742

03100 061055

03101 031743

03102 062531

03103 011743

03104 067063

03105 025752

03106 035400

03107 062531

03110 070306

03111 071711

03112 070153

03113 021750

03114 170402

03115 021750

03116 067101

TBS6

TBSR1

TBSRH

TBS5

TBS2

TBS4

JSM GTMP2

RAR 7

SLA TBS5

SAR 9

AND .63

JMP RETN2

LDA TMP5

STA TMP4

STB TMP2

LDB BADDR

STB TMP6A

JMP TBS2

DSZ TMP2

JSM GNEXT

STA TMP3

JSM GTMP2

CPA TMP3

JMP TBS6

LDB TMP6A

STB BADDR

JSM GTMP2

RAR 7

SLA *-2

SAP TBS4

LDA TMP4

RET

LDA TMP4

JMP TBS2

OP CODE?

YES, RIGHT JUSTIFY

SAVE INPUT CHAR

INITIALIZE TABLE POINTER

SAVE INPUT POINTER

FETCH NEXT INPUT CHAR

STORE CHARACTER

CHARACTERS MATCH?

YES, CONTINUE

NO, RESTORE BADDR

CHAR IS OP CODE?

LAST ENTRY?

RESTORE FIRST CHARACTER

RETURN TO P+1

CONTINUE

PAGE 42

KEYBOARD ENTRY AND DISPLAY

2037*
 2038* CONSTANTS, TEMPORARIES, ETC.
 2039* USED IN ABOVE ROUTINES
 2040*
 2041*
 2042 03117 001742 TMP2A DEF TMP+2
 2043 03120 006006 LISTR DEF LIST1
 2044 03121 000200 B200 OCT 200
 2045 03122 001704 BKPTA DEF BKPTR
 2046 03123 001621 ENDW DEF RBUF+40
 2047 03124 000033 B33 OCT 33
 2048 03125 000176 B176 OCT 176
 2049 03126 000137 B137 OCT 137
 2050 03127 000163 DELIN OCT 163
 2051 03130 003321 IOEND DEF IOBUF+IOBUF+81

PAGE 43

DEFINE-A-KEY-ROUTINES

2053*
 2054* DEFINE-A-KEY-ROUTINES
 2055*
 2056* THE FIRST WORD OF EACH DEFINITION CONTAINS THE
 2057* KEYCODE OF THAT KEY. THE SECOND WORD CONTAINS THE LENGTH
 2058* OF THE DEFINITION PLUS 1. WHILE IN DEFINITION MODE
 2059* "UKPTR" POINTS TO THE LENGTH WORD.
 2060*
 2061 03131 060626 KEYPR JSM SAVBP SAVE BADDR
 2062 03132 020212 LDA SLBF
 2063 03133 031400 STA BADDR
 2064 03134 061055 JSM GNEXT SET UP FOR TABLE SEARCH
 2065 03135 027152 LDB KTBA
 2066 03136 160175 JSM TSRHA,I SEARCH TABLE
 2067 03137 067153 JMP RCAL NOT FOUND
 2068 03140 074742 SBR 16
 2069 03141 035432 STB DRQST
 2070 03142 050024 AND .7
 2071 03143 003145 ADA JTBA A-REG HAS OPCODE
 2072 03144 070737 JMP A,I
 2073*
 2074 03145 103145 JTBA DEF JTBL-1,I
 2075 03146 003253 JTBL DEF XRUN RUN
 2076 03147 003310 DEF XSCR SCRATCH
 2077 03150 003330 DEF XLST LIST
 2078 03151 003261 DEF XKEY FETCH
 2079*
 2080 03152 007155 KTBA DEF KTBL+KTBL+1

PAGE 44

DEFINE-A-KEY-ROUTINES

2082	03153	060633	RCAL	JSM	RESB1	RESTORE BADDR
2083	03154	063346		JSM	HUNT	FIND THE LOCATION
2084	03155	160206	KERR	JSM	ERRA,1	NOT FOUND, ERROR
2085	03156	074517		LDA	B,1	
2086	03157	031466		STA	LOCLI	SAVE LENGTH WORD
2087	03160	070744		SAL	1	
2088	03161	000136		ADA	M5	ASSUME ODD COUNT
2089	03162	070113		SAP	*+2	NULL DEFINITION?
2090	03163	066045		JMP	REED6	YES, IGNORE
2091	03164	031473		STA	CCNT	SAVE CHARACTER COUNT
2092	03165	035470		STB	L2TI	SAVE LOCATION
2093	03166	074070		SIB	*+1	
2094	03167	074517		LDA	B,1	LOOK AT LINE #
2095	03170	074070		SIB	*+1	
2096	03171	070110		SZA	*+2	IS THIS A TYPING AID?
2097	03172	067222		JMP	RFNX	NO
2098	03173	074744		SBL	1	YES
2099	03174	035740		STB	TMP	SAVE CHARACTER POINTER
2100	03175	074002		SBR	1	
2101	03176	005466		ADB	LOCLI	
2102	03177	004134		ADB	M3	
2103	03200	074517		LDA	B,1	LOOK AT LAST WORD
2104	03201	050105		AND	B377	
2105	03202	070450		SZA	*+9	
2106*						
2107	03203	045473		ISZ	CCNT	CORRECT COUNT
2108	03204	010100		CPA	ASTR1	CHECK FOR *
2109	03205	067207		JMP	*+2	
2110	03206	066045		JMP	REED6	
2111	03207	074517		LDA	B,1	
2112	03210	050157		AND	M256	
2113	03211	040027		IOR	AIEX1	IEX CODE
2114	03212	067220		JMP	*+6	
2115*						
2116	03213	074517		LDA	B,1	
2117	03214	013362		CPA	ASTR2	CHECK FOR*
2118	03215	067217		JMP	*+2	
2119	03216	066045		JMP	REED6	
2120	03217	023363		LDA	AIEX2	IEX CODE
2121	03220	074557		STA	B,1	
2122	03221	066045		JMP	REED6	
2123*						
2124	03222	074517	RFNX	LDA	B,1	CHECK FOR LEGITIMATE FUNCTION
2125	03223	050065		AND	K1	(076000)
2126	03224	010004		CPA	K2	(010000)
2127	03225	076130		RIB	*+2	
2128	03226	067246		JMP	RRUN	NOT DEF
2129	03227	074517		LDA	B,1	
2130	03230	053357		AND	K3	(176037)
2131	03231	013360		CPA	K4	(012037)
2132	03232	067234		JMP	*+2	
2133	03233	067246		JMP	RRUN	NOT DEF
2134	03234	020162		LDA	FN	
2135	03235	031754		STA	AR2	"FN"
2136	03236	074517		LDA	B,1	
2137	03237	070202		SAR	5	EXTRACT THE LETTER

PAGE 45

DEFINE-A-KEY-ROUTINES

2138	03240	050040		AND	.31	
2139	03241	040101		IOR	B100	
2140	03242	070404		SAL	8	
2141	03243	031755		STA	AR2+1	"X"
2142	03244	027361		LDB	ALCL1	GET POINTER
2143	03245	066246		JMP	RFX1	SET UP COUNT
2144*						
2145	03246	061225	RRUN	JSM	LDEF	LEAVE DEF MODE
2146	03247	025470		LDB	L2T1	YES, CAN RUN
2147	03250	063336		JSM	SUMP	
2148	03251	070742		SAR	16	
2149	03252	066157		JMP	CONTK	CONTINUE FROM BEGINNING OF KEY
2150*						
2151	03253	061225	XRUN	JSM	LDEF	INSURE NOT IN KEY MODE
2152	03254	063346		JSM	HUNT	
2153	03255	067155		JMP	KERR	KEY UNDEFINED
2154	03256	063336		JSM	SUMP	SET UP PROGRAM POINTERS
2155	03257	070742		SAR	16	
2156	03260	164171		JMP	RUNA,I	RUN
2157*						
2158*						
2159	03261	061225	XKEY	JSM	LDEF	INSURE NOT IN DEFINITION MODE
2160	03262	063346		JSM	HUNT	FIND THE LOCATION
2161	03263	063270		JSM	NEWK	NOT FOUND, NEW KEY REQUIRED
2162	03264	063336		JSM	SUMP	
2163	03265	060651		JSM	INITS	INITIALIZE STACKS
2164	03266	070742		SAR	16	
2165	03267	066254		JMP	EFET+1	FETCH FIRST LINE
2166*						
2167	03270	160242	NEWK	JSM	DCOMA,I	DECOMPILE PROGRAM
2168	03271	021405		LDA	FWUP	ADD A NEW KEY TO DEFINITION AREA
2169	03272	031476		STA	TEMP3	
2170	03273	020017		LDA	.2	
2171	03274	160250		JSM	OVCHA,I	CHECK FOR OVERFLOW
2172	03275	061121		JSM	MVTOH	MOVE CORE
2173	03276	021751		LDA	TMPS	
2174	03277	131405		STA	FWUP,I	STORE NEW KEY
2175	03300	045405		ISZ	FWUP	
2176	03301	025405		LDB	FWUP	
2177	03302	045405		ISZ	FWUP	UPDATE FWUP
2178	03303	020012		LDA	.1	
2179	03304	074557		STA	B,I	INITIALIZE LENGTH WORD
2180	03305	045424		ISZ	PBUFF	
2181	03306	045424	NEWK1	ISZ	PBUFF	UPDATE PBUFF
2182	03307	170402		RET		

PAGE 46

DEFINE-A-KEY-ROUTINES

```

2184*
2185*  SCRATCH A KEY
2186*
2187  03310  061225  XSCR  JSM  LDEF      INSURE NOT IN DEFINITION MODE
2188  03311  160242      JSM  DCOMA,IDE_COMPILE
2189  03312  063346      JSM  HUNT      FIND THE LOCATION
2190  03313  164174      JMP  RDYDA,I  NOT FOUND, IGNORE
2191  03314  074117      LDA  B
2192  03315  070437      ADB  A,I      END OF AREA
2193  03316  000132      ADA  M1      START OF AREA
2194  03317  031477  XSCR1 STA  TEMP4
2195  03320  160251      JSM  CLPIA,I  CLOSE UP AREA
2196  03321  021405      LDA  FWUP
2197  03322  074017      ADA  B
2198  03323  031405      STA  FWUP      UPDATE FWUP
2199  03324  005424      ADB  PBUFF
2200  03325  035424      STB  PBUFF      UPDATE PBUFF
2201  03326  060651      JSM  INITS    INITIALIZE STACKS
2202  03327  164174      JMP  RDYDA,I  RETURN TO MONITOR
2203*
2204*  LIST A DEFINITION
2205*
2206  03330  063346  XLST  JSM  HUNT      FIND THE LOCATION
2207  03331  067615      JMP  XLST1+1  NOT FOUND
2208  03332  074117      LDA  B
2209  03333  070437      ADB  A,I      END OF AREA
2210  03334  070070      SIA  *+1      START OF AREA
2211  03335  067614      JMP  XLST1  LIST DEFINITION
2212*
2213*  SET UP MISCELLANEOUS POINTERS
2214*
2215  03336  021424  SUMP  LDA  PBUFF
2216  03337  031425      STA  PBUFO      SAVE OLD PBUFF
2217  03340  035422      STB  UKPTR
2218  03341  035424      STB  PBUFF
2219  03342  074517      LDA  B,I
2220  03343  070037      ADB  A
2221  03344  035423      STB  PBPTR      ESTABLISH NEW PBPTR
2222  03345  067306      JMP  NEWK1

```

PAGE 47

DEFINE-A-KEY-ROUTINES

```

2224*
2225* SUBROUTINE TO FIND A DEFINED KEY AREA
2226*
2227* RETURN TO P+1 IF NOT FOUND
2228* RETURN TO P+2 IF FOUND WITH B-REG = ADDRESS OF LENGTH WORD
2229*
2230 003346 024166 HUNT LDB FWAM START SEARCHING AT FWAM
2231 003347 015405 HUN1 CPB FWUP
2232 003350 170402 RET
2233 003351 074517 LDA B,I UNDEFINED
2234 003352 074070 SIB *+1
2235 003353 011751 CPA TMP5 EXTRACT THIS KEYCODE
2236 003354 064357 JMP RETN2 AND COMPARE WITH DESIRED
2237 003355 074437 ADB B,I FOUND IT
2238 003356 067347 JMP HUN1
2239*
2240* CONSTANTS
2241*
2242 000065 K1 EQU OPMSK (076000)
2243 000004 K2 EQU B10K (010000)
2244 003357 176037 K3 OCT 176037
2245 003360 012037 K4 OCT 012037
2246 003361 003730 ALCL1 DEF AR2+AR2
2247 001000 ASTR1 EQU B52
2248 003362 025000 ASTR2 OCT 025000
2249 000027 AIEX1 EQU .11
2250 003363 005400 AIEX2 OCT 005400

```

PAGE 48

SYSTEM TABLES

2252*

2253* NON-FORMULA OPERATORS

2254*

2255	03364	052110	FOP13	ASC	2,THEN
	03365	042516			
2256	03366	150117		OCT	150117 THEN; 20
2257	03367	043321		OCT	043321 OF; 21
2258	03370	043203	FTABL	OCT	43203 F; 3
2259	03371	042704		OCT	42704 E; 4; EOT
2260	03372	041206		OCT	41206 B; 6
2261	03373	054207		OCT	54207 X; 7
2262	03374	027710		OCT	27710 /; 10; EOT
2263	03375	027311		OCT	27311 ;; 11; EOT
2264	03376	051524	FOPST	ASC	2,STEP
	03377	042520			
2265	03400	156524		OCT	156524 STEP 35
2266	03401	047736		OCT	047736 TO; 36
2267	03402	047522	OR	ASC	1,OR
2268	03403	113101		OCT	113101 OR; 26
2269	03404	047104		ASC	1,ND
2270	03405	153516		OCT	153516 AND; 27
2271	03406	047524		ASC	1,OT
2272	03407	154076	GTE	OCT	154076 NOT; 30
2273	03410	036631		OCT	036631 >=; 31
2274	03411	036075		ASC	1,<=
2275	03412	115074		OCT	115074 <=; 32
2276	03413	037320		OCT	037320 <>; 20

2277*

SYSTEM COMMAND JUMP TABLE

2278*

2279*

2280	03414	004322		ABS	FLT-*
2281	03415	004322		ABS	FIXED-*
2282	03416	000437		ABS	PTAPE-*
2283	03417	176534	LBL1	ABS	CONT-*
2284	03420	001237		ABS	DELET-*
2285	03421	001244		ABS	RENUM-*
2286	03422	004335		ABS	NORML-*
2287	03423	004331		ABS	TRACE-*
2288	03424	000151		ABS	AUTO-*
2289	03425	176626		ABS	EFET-*
2290	03426	000155		ABS	LISTC-*
2291	03427	001212		ABS	SCRAT-*
2292	03430	001311		ABS	RUN-*

2293*

SYSTEM COMMAND NAME TABLE

2294*

2295*

2296	03431	051105	CMD13	ASC	1,RE
2297	03432	047210		OCT	47210 REN; 001000
2298	03433	042105		ASC	1,DE
2299	03434	046271		OCT	046271 DELETE; 111001
2300	03435	050124		ASC	1,PT
2301	03436	040653		OCT	40653 PTAPE; 1011011
2302	03437	043111		ASC	1,FI
2303	03440	054214		OCT	54214 FIXED; 001100
2304	03441	043114		ASC	1,FL
2305	03442	052215		OCT	52215 FLOAT; 001101

170

158

1,444,141

158

PAGE 49

SYSTEM TABLES

2306	03443	052122	ASC	2,TRAC	
	03444	040503			
2307	03445	042666	OCT	42666	TRACE; 110110
2308	03446	047117	ASC	3,NORMAL	
	03447	051115			
	03450	040514			
2309	03451	103503	OCT	103503	NORMAL; 000111
2310	03452	047516	ASC	1,ON	
2311	03453	052252	OCT	52252	CONT; 101010
2312	03454	051524	ASC	2,STOP	
	03455	047520			
2313	03456	137501	OCT	137501	STOP; 111111 (NO JUMP)
2314	03457	052524	ASC	1,UT	
2315	03460	047605	OCT	47605	AUTO 000101
2316	03461	051524	ASC	2,STOR	
	03462	047522			
2317	03463	042600	OCT	42600	STORE; 000000 (NO JUMP)
2318	03464	046117	ASC	2,LOAD	
	03465	040504			
2319	03466	110114	OCT	110114	LOAD; 010000 (NO JUMP)
2320	03467	044523	ASC	1,IS	
2321	03470	052203	OCT	52203	LIST; 000011
2322	03471	051125	ASC	1,RU	
2323	03472	047241	OCT	047241	RUN; 100001
2324	03473	051503	ASC	3,SCRATC	
	03474	051101			
	03475	052103			
2325	03476	044202	OCT	44202	SCRATCH; 000010
2326	03477	043105	ASC	2,FETC	
	03500	052103			
2327	03501	044344	OCT	44344	FETCH; 100100

PAGE 50

SYNTAX

2329*

2330* DATA STATEMENT SYNTAX

2331*

2332	03502	045706	DATAS	ISZ	SBPTR	
2333	03503	061156		JSM	CONST	LOOK FOR A NUMBER
2334	03504	160206	NDATA	JSM	ERRA,I	NOT FOUND
2335	03505	160244		JSM	NUMOA,I	FOUND; STORE IN SYNTAX BUFFER
2336	03506	060504	DATA2	JSM	COMCK	COMMA?
2337	03507	065151		JMP	EOST	NO
2338	03510	067502		JMP	DATAS	

2339*

2340* FORMAT STATEMENT SYNTAX

2341*

2342	03511	061055	FMTS	JSM	GNEXT	GET FIRST CHAR OF NEXT SPEC
2343	03512	060750		JSM	DIGCK	DIGIT?
2344	03513	067521		JMP	FMTF	NO
2345	03514	035724		STB	GFLAG	YES, SET FOR ERROR ON OVERFLOW
2346	03515	055400		DSZ	BADDR	FETCH AND
2347	03516	024161		LDB	MAXSN	STORE
2348	03517	060666		JSM	PGINT	REPEAT FIELD
2349	03520	067522		JMP	*+2	
2350	03521	045706	FMTF	ISZ	SBPTR	MOVE TO NEW WORD
2351	03522	027552		LDB	FADD	LOOK FOR
2352	03523	060546		JSM	STEXP	"F" OR "E"
2353	03524	067537		JMP	FMTB	NOT FOUND
2354	03525	024161		LDB	MAXSN	FOUND; FETCH AND
2355	03526	060666		JSM	PGINT	STORE FIELD WIDTH
2356	03527	027553		LDB	ADD	LOOK FOR
2357	03530	060546		JSM	STEXP	
2358	03531	160206	FMTER	JSM	ERRA,I	NOT FOUND
2359	03532	024161	FMTD	LDB	MAXSN	
2360	03533	061131		JSM	PGIN0	FETCH DIGIT FIELD, ALLOW 0
2361	03534	061153	FMTC	JSM	EOLCK	EOL?
2362	03535	060510		JSM	COMCE	NO, DEMAND A COMMA
2363	03536	067511		JMP	FMTS	GET NEXT SPEC
2364	03537	027554	FMTB	LDB	BADD	
2365	03540	060546		JSM	STEXP	SEARCH FOR "B", "X", OR "/"
2366	03541	067545		JMP	FMTQT	NOT FOUND
2367	03542	061055		JSM	GNEXT	FOUND
2368	03543	045706		ISZ	SBPTR	
2369	03544	067534		JMP	FMTC	DEMAND COMMA OR EOL
2370	03545	010043	FMTQT	CPA	B42	QUOTE?
2371	03546	067550		JMP	*+2	
2372	03547	067531		JMP	FMTER	NO
2373	03550	160245		JSM	GTSTA,I	YES, GET QUOTE FIELD
2374	03551	067534		JMP	FMTC	
2375	03552	006760	FADD	DEF	FTABL+FTABL	
2376	03553	006772	ADD	DEF	FTABL+FTABL+10	
2377	03554	006764	BADD	DEF	FTABL+FTABL+4	

2378*

2379* GOTO AND GOSUB STATEMENT SYNTAX

2380*

2381	03555	060626	GOTOS	JSM	SAVBP	SAVE BADDR AND SBPTR
2382	03556	055724		DSZ	GFLAG	SET TO RETURN ON ERROR FROM INTCK
2383	03557	060664		JSM	LNUM	LOOK FOR A LINE NUMBER
2384	03560	067572		JMP	GOTO3	FOUND

172

160

1,444,141

160

PAGE 51

SYNTAX

2385	03561	060640	GOTO1	JSM	RESBP	RESTORE POINTERS
2386	03562	160202		JSM	FSCA,I	FETCH FORMULA
2387	03563	027574		LDB	OFA	
2388	03564	060546		JSM	STEXP	STORE "OF" AS EXPANDED OPERATOR
2389	03565	160206	MISOF	JSM	ERRA,I	NOT FOUND
2390	03566	060664	GOTO2	JSM	LNUM	FETCH LINE NUMBER
2391	03567	061153		JSM	EOLCK	EOL?
2392	03570	060510		JSM	COMCE	DEMAND A COMMA
2393	03571	067566		JMP	GOTO2	
2394	03572	061153	GOTO3	JSM	EOLCK	EOL?
2395	03573	067561		JMP	GOTO1	NO, TRY FOR A FORMULA INSTEAD
2396	03574	006755	OFA	DEF	FOP13+	FOP13+5
2397*						
2398*	AUTO SYSTEM COMMAND					
2399*						
2400	03575	063647	AUTO	JSM	GETLN	GET OPTIONAL PARAMETERS
2401	03576	020061		LDA	.10	
2402	03577	024061		LDB	.10	
2403	03600	031427		STA	LNUM	SET INITIAL LINE NUMBER
2404	03601	035430		STB	LNINC	SET INCREMENT
2405	03602	164173		JMP	PEXMA,I	
2406*						
2407*	LIST EXECUTION					
2408*						
2409	03603	061055	LISTC	JSM	GNEXT	GET NEXT CHARACTER
2410	03604	013627		CPA	B43	"#"?
2411	03605	067623		JMP	LSTC2	YES
2412	03606	055400		DSZ	BADDR	RESET POINTER
2413	03607	063647	LSTC1	JSM	GETLN	GET OPTIONAL LINE #'S
2414	03610	070742		SAR	16	
2415	03611	024056		LDB	.1E4	
2416	03612	060730		JSM	GETAD	GET ADDRESSES
2417	03613	021477		LDA	TEMP4	
2418	03614	160122	XLST1	JSM	LISTA,I	LIST
2419	03615	160170		JSM	CLERA,I	CLEAR BUFFER
2420	03616	025412		LDB	LSTPT	
2421	03617	074076		TCB		
2422	03620	005413		ADB	HSTPT	COMPUTE WORDS BETWEEN STACKS
2423	03621	160210		JSM	OUTIA,I	PUT IN BUFFER
2424	03622	164173		JMP	PEXMA,I	
2425	03623	060664	LSTC2	JSM	LNUM	FETCH SELECT CODE
2426	03624	074204		SBL	12	
2427	03625	035447		STB	SLCOD	
2428	03626	067607		JMP	LSTC1	
2429	03627	000043	B43	OCT	43	
2430*						
2431*	STOP STATEMENT SYNTAX					
2432*						
2433	03630	065147		JMP	ENDS	
2434	03631	021416	STOPS	LDA	MODE	
2435	03632	071711		SLA	STOPS-1	SKIP IF PROG
2436	03633	063647		JSM	GETLN	GET PARAMETERS
2437	03634	020056		LDA	.1E4	
2438	03635	024056		LDB	.1E4	SET DEFAULT PARAMETERS
2439	03636	060741		JSM	FND-1	FIND ADDRESS OF FIRST
2440	03637	015423		CPB	PBPTR	PAST END OF PROG?

PAGE 52

SYNTAX

```

2441 03640 074742 SBR 16 YES, CLEAR FLAG
2442 03641 035436 STB HFLG1
2443 03642 060740 JSM FND-2 FIND ADDRESS OF SECOND
2444 03643 015423 CPB PBPTR
2445 03644 074742 SBR 16
2446 03645 035437 STB HFLG1+1
2447 03646 164174 JMP RDYDA,I
2448*
2449*
2450* SUBROUTINE TO BUILD AND STORE N>0 LINE NUMBERS
2451* RETURN TO: P+1*NONE GIVEN
2452* P+2:1 GIVEN (A)=NUMBER
2453* P+3:2 GIVEN (A)=FIRST, (B)=SECOND
2454* NUMBERS ARE STORED SEPARATED BY COMMAS IN S-BUFFER
2455*
2456 03647 061066 GETLN JSM GETSK GET NEXT CHAR
2457 03650 170402 RET NONE GIVEN
2458 03651 060750 JSM DIGCK DIGIT?
2459 03652 067647 JMP GETLN NO, IGNORE IT
2460 03653 060663 JSM LNUM-1 BUILD FIRST INTEGER
2461 03654 035720 STB LASTN SAVE 1ST #
2462 03655 010031 CPA EOL END OF LINE?
2463 03656 067662 JMP GETL2 YES, RETURN TO P+2
2464 03657 060365 JSM RET2 INCREMENT RETURN ADDRESS
2465 03660 060510 JSM COMCE FIND AND RECORD A COMMA
2466 03661 060664 JSM LNUM GET NEXT LINE NUMBER
2467 03662 021720 GETL2 LDA LASTN MOVE TO (A)
2468 03663 064357 GETLR JMP RETN2 RETURN TO P+2 OR P+3
2469*
2470 03664 000000 BSS 1 *****CHECKSUM*****
2471*
2472* CLEAR ROUTINE
2473*
2474 03665 070742 ECLR SAR 16
2475 03666 031430 STA LNINC CLEAR AUTO MODE
2476 03667 170402 RET RETURN TO P+1
2477* 3677 IS LAST
2478* MATH TABLES HERE

```

PAGE 53

9830—MONITOR

2480*					
2481	04000		ORG 4000B		
2482*					
2483	04000	020116	ENTRY LDA QTOP	CLEAR THE BASE PAGE	
2484	04001	170000	CLR		
2485	04002	000135	ADA M4		
2486	04003	073710	RZA *-2		
2487	04004	024066	LDB INF	DETERMINE THE	
2488	04005	004160	MAWCK ADB M1024		
2489	04006	074577	STB B,I		
2490	04007	074477	CPB B,I	LAST WORD OF R/W	
2491	04010	066012	JMP *-2	YES, IT WAS	
2492	04011	066005	JMP MAWCK	CHECK NEXT PAGE	
2493*					
2494*					
2495	04012	074557	STA B,I	INITIALIZE LAST WORD	
2496	04013	035403	STB MAW		
2497	04014	004132	ADB M1	SET THE POINTER	
2498	04015	035404	STB LWAM	TO THE LAST AVAILABLE WORD	
2499	04016	024116	LDB REED		
2500	04017	035462	STB REEDL	INITIALIZE KEYBOARD LINK	
2501	04020	024207	LDB INTEA		
2502	04021	035463	STB INTEL	INITIALIZE INTERRUPT LINK	
2503	04022	020026	LDA .9		
2504	04023	031452	STA CPLAC	SET DECIMAL SETTING TO 9	
2505	04024	031451	STA CTYPE	SET MODE TO "FLOAT"	
2506	04025	045434	ISZ UNITS	INITIALIZE TO RADIAN	
2507	04026	045450	ISZ PMODE	INITIALIZE TO NORMAL	
2508	04027	020166	KILLK LDA FWAM	INITIALIZE START OF	
2509	04030	031405	STA FWUP	USER PROGRAM(SCRATCH KEYS)	
2510	04031	020112	KILLP LDA RSTKA	INITIALIZE RETURN	
2511	04032	031777	STA RSPTR	POINTER	
2512	04033	062734	JSM SCRP	INITIALIZE PROGRAM POINTERS	
2513	04034	061233	JSM LDEF+6	RESET USER KEY POINTER	
2514	04035	060645	KILLV JSM MINIT	INITIALIZE SYMBOL TABLE	
2515	04036	060651	JSM INITS	INITIALIZE STACKS AND FLAGS	
2516	04037	070742	RDYPT SAR 16		
2517	04040	031433	STA TFLAG	CLEAR PTAPE FLAG	
2518	04041	031430	STA LNINC	SET TO MANUAL	
2519*					
2520	04042	160170	JSM CLERA,I	CLEAR DISPLAY	
2521	04043	031622	STA TERM	RESET CRLF FLAG	
2522	04044	070742	PEXMK SAR 16		
2523	04045	031701	STA NDISP		
2524	04046	031702	STA NPRIN		
2525	04047	021432	PEXM2 LDA DRQST		
2526	04050	070510	SZA PEXM3	SKIP IF NOT DATA REQUEST	
2527	04051	020012	LDA .1		
2528	04052	026267	LDB QMRKA	PRINT A ?	
2529	04053	061330	JSM .NPWR		
2530	04054	066100	JMP PEXM1		
2531*	PTAPE SYSTEM COMMAND				
2532	04055	031433	PTAPE STA TFLAG	STORE DEVICE CODE	
2533	04056	072110	R2A *-2		
2534	04057	064726	JMP SYE25	SELECT CODE NOT GIVEN	
2535	04060	020132	LDA M1		

2536	04061	031742		STA	TMP2	SET FLAG FOR LEADER
2537	04062	070742	PEXM3	SAR	16	
2538	04063	031442		STA	STPFL	CLEAR THE STEP-STOP FLAG
2539	04064	020127		LDA	B170K	
2540	04065	031447		STA	SLCOD	SET TO STANDARD PRINTER
2541	04066	021400		LDA	BADDR	
2542	04067	010212		CPA	.SLBF	BLANK SCREEN?
2543	04070	066072		JMP	*+2	YES
2544	04071	066100		JMP	PEXM1	
2545	04072	026264		LDB	LTADD	GET ADDRESS OF LAZY-T
2546	04073	021422		LDA	UKPTR	
2547	04074	070170		SIA	*+3	SKIP IF NOT KEY MODE (A)=1
2548	04075	004134		ADB	M3	GET ADDRESS OF KEY-LAZY-T
2549	04076	020021		LDA	.4	
2550	04077	061335		JSM	NPWRT	WRITE ON DISPLAY
2551*						
2552	04100	024112	PEXM1	LDB	RSTKA	
2553	04101	035777		STB	RSPTR	RESET STACK
2554	04102	020041	ALNUM	LDA	.32	
2555	04103	031707		STA	BLANK	SET TO IGNOR BLANKS
2556	04104	021430		LDA	LNINC	
2557	04105	070410		SZA	GTRC1	SKIP IF MANUAL
2558	04106	021701		LDA	NDISP	
2559	04107	072350		RZA	GTRC1+1	SKIP IF MESSAGE ON SCREEN
2560	04110	160170		JSM	CLERA,I	CLEAR OFF LAZY-T
2561	04111	025427		LDB	.LNUM	
2562	04112	160210		JSM	OUTIA,I	PUT LINE # IN BUFFER
2563	04113	045400		ISZ	BADDR	OUTPUT A SPACE
2564	04114	066116		JMP	*+2	
2565	04115	031427	GTRC1	STA	.LNUM	SET LINE # IF MANUAL
2566	04116	161462	GTRCD	JSM	REEDL,I	READ RECORD
2567	04117	066042		JMP	PEXMK-2	NULL RECORD
2568	04120	061062		JSM	GFRST	GET FIRST CHAR IN (A)
2569	04121	066042		JMP	PEXMK-2	NULL RECORD
2570	04122	074742		SBR	16	
2571	04123	035724		STB	GFLAG	SET FLAG FOR INTEGER ERROR
2572	04124	025432		LDB	DRQST	
2573	04125	074730		SIB	CKRCD	SKIP IF NOT DATA REQUEST
2574	04126	035642		STB	SAVE	SAVE RETURN ADDRESS
2575	04127	074742		SBR	16	
2576	04130	035432		STB	DRQST	CLEAR THE INPUT FLAG
2577	04131	035416		STB	MODE	SET TO PROGRAM EXECUTION MODE
2578	04132	160237		JSM	SALTA,I	STATEMENT GIVEN?
2579	04133	066260		JMP	GTRC2	NO
2580	04134	020127	GTRC3	LDA	B170K	YES
2581	04135	031447		STA	SLCOD	SET TO STANDARD PRINTER
2582	04136	022270		LDA	CALSN	
2583	04137	031416		STA	MODE	SET TO CALC SYNTAX MODE
2584	04140	061062		JSM	GFRST	
2585	04141	000000		NOP		*** CHECKSUM
2586	04142	066154		JMP	SYNTAX	
2587	04143	025416	CKRCD	LDB	MODE	
2588	04144	074111		SLB	*+2	SKIP IF EOL
2589	04145	066154		JMP	SYNTAX	EXECUTE: DONT ALLOW LINE #
2590	04146	010100		CPA	STAR	FIRST CHAR AN "***"
2591	04147	066251		JMP	UKDEF	YES

176

164

1,444,141

164

PAGE 55

9830-MONITOR

2592	04150	060675	JSM	INTCK-1	BUILD A LINE #
2593	04151	035427	STB	LNUM	STORE LINE #
2594*			(A)	CONTAINS NEXT CHAR	
2595	04152	010031	CPA	EOL	END OF LINE?
2596	04153	066500	JMP	DELST	YES, DELETE LINE FROM CODE
2597*					
2598	04154	025426	SYNTAX	LDB	PBPTO
2599	04155	015424	CPB	PBUFF	GET MAINLINE END POINTER
2600	04156	062734	JSM	SCRP	NULL PROGRAM?
					YES, INSURE NO SPURIOUS COMMON EXISTS
2601	04157	024026	LDB	9	SET STORE FLAG FALSE
2602	04160	035710	STB	MSFLG	(DONT RECOGNISE ASSIGNMENT)
2603	04161	025413	LDB	HSTPT	INITIALIZE
2604	04162	004132	ADB	M1	THE S-STACK
2605	04163	035472	STB	TEMPS	POINTER
2606	04164	074742	SER	16	
2607	04165	035711	STB	DFLAG	CLEAR DEFINE FLAG
2608	04166	035714	STB	SFLAG	
2609	04167	160237	STMT	JSM	SEARCH TABLES FOR STATEMENT
2610	04170	066201	JMP	SALTA,I	NOT FOUND
2611	04171	045706	ISZ	CMND	
2612	04172	135706	STB	SBPTR	
2613	04173	025416	STB	SBPTR,I	STORE THE PAGE
2614	04174	074211	LDB	MODE	
2615	04175	070304	SLB	STMT1	SKIP IF PROGRAM MODE
2616	04176	070112	SAL	10	SHIFT OP CODE
2617	04177	062360	NCALC	SAM	SKIP IF STMT TYPE IS ALLOWED
2618	04200	165473	STMT1	JSM	STMT NOT ALLOWED IN CALC
2619	04201	024132	CMND	JMP	BRANCH TO CODE FOR STATEMENT
2620	04202	160237	LDB	M1	SEARCH TABLES FOR
2621	04203	066221	JSM	SALTA,I	SYSTEM COMMAND
2622	04204	074742	CMNDF	JMP	NOT FOUND
2623	04205	035431	SBR	16	
2624	04206	035430	STB	CLINE	RESET POINTER FOR LIST
2625	04207	035416	STB	LNINC	RESET AUTO
2626	04210	035442	STB	MODE	SET TO PROG
2627	04211	070142	STB	STPFL	CLEAR "STEP" FLAG
2628	04212	070210	SAR	4	GET # OF PARAMETERS ALLOWED
2629	04213	162733	SZA	JCMND	BRANCH IMMEDIATELY
2630	04214	070742	JSM	GETLA,I	GET LINE # PARAMETERS
2631	04215	024056	SAR	16	NONE GIVEN, ASSUME 0
2632	04216	161473	JCMND	LDB	ONLY ONE GIVEN; ASSUME INFINITY
2633*			JSM	TEMP,I	BRANCH TO COMMAND EXECUTION
2634*					ALL COMMAND JUMP TO PEXMK OR RDYPT EXCEPT FETCH
2635*					WHICH JUMPS TO READ; THE FOLLOWING TWO INSTRUCTIONS
2636	04217	066042	JMP	PEXMK-2	
2637	04220	066120	JMP	GTRCD+2	
2638*					ARE RETURN EXITS FOR READ
2639	04221	070742	ILET	SAR	16
2640	04222	066164	JMP	STMT-3	GET SPEICAL CHAR FOR IMPLIED LET
2641*					FIND BLOCK FOR IMPLIED LET
2642	04223	055400	ILETs	DSZ	BADDR
2643	04224	061055	JSM	GNEXT	
2644	04225	055400	DSZ	BADDR	RECALL CHARACTER
2645	04226	060752	JSM	LETCK	
2646	04227	066234	JMP	ILET1+1	LETTER?
2647	04230	060626	JSM	SAVBP	NO, TRY IMPLIED DISP
					SAVE POINTERS

PAGE 56

9830—MONITOR

2648	04231	062274		JSM	LETSN	TRY "LET" SNTAX
2649	04232	066244		JMP	NSTMT+1	OK, CHECK FOR EOL
2650	04233	060640	ILET1	JSM	RESBP	RESTORE POINTERS
2651	04234	020125		LDA	PRTOP	STORE CODE FOR
2652	04235	055706		DSZ	SBPTR	
2653	04236	131706		STA	SBPTR,I	IMPLIED LET
2654	04237	045706		ISZ	SBPTR	
2655	04240	025416		LDB	MODE	
2656	04241	074111		SLB	NSTMT	SKIP IF PROG
2657	04242	062315		JSM	PRNTS	TRY "PRINT" SYNTAX
2658	04243	062360	NSTMT	JSM	ERROR	NO STATEMENT TYPE CAN BE FOUND
2659	04244	061153		JSM	EOLCK	END OF LINE?
2660	04245	021416		LDA	MODE	NO
2661	04246	070111		SLA	*+2	
2662	04247	066233		JMP	ILET1	TRY PRINT SYNTAX IF CALC MODE
2663	04250	065152		JMP	NOEOF	
2664*						
2665	04251	021422	UKDEF	LDA	UKPTR	IN KEY DEFINITION MODE?
2666	04252	072110		RZA	*+2	
2667	04253	160206	UKERR	JSM	ERRA,I	NO, DEFINITION NOT ALLOWED
2668	04254	074742		SBR	16	
2669	04255	135706		STB	SBPTR,I	SET A 0 LINE NUMBER
2670	04256	160253		JSM	CHRSA,I	ACCEPT AN ASCII STRING
2671	04257	066520		JMP	ACTKY	ACCEPT DEFINITION
2672*						
2673	04260	024132	GTRC2	LDB	M1	
2674	04261	160237		JSM	SALTA,I	SYSTEM COMMAND IN "INPUT"?
2675	04262	165642		JMP	SAVE,I	NO, RETURN DATA TO INPUT STMT
2676	04263	066134		JMP	GTRC3	YES, ABORT INPUT
2677*						
2678	04264	010555	LTADD	DEF	*+*+*5	
2679	04265	045505		ASC	1,KE	
2680	04266	054437		OCT	54437	Y,LAZY-T
2681	04267	000131	QMRKA	DEF	.63+.63+1	
2682	04270	100001	CALSN	OCT	100001	
2683	00116		IMPOP	EQU	QTOP	OP CODE 1
2684	00125		PRTOP	EQU	LBOP	OP CODE 24
2685	00100		STAR	EQU	B52	
2686	00123		STDPG	EQU	RBOP	
2687*						
2688*						
2689*						
2690	04271	062274	LETS	JSM	LETSN	
2691	04272	065151		JMP	EOST	
2692	04273	160206	SYNE2	JSM	ERRA,I	
2693*						
2694*						
2695*						
2696*						
2697*						
2698	04274	055710	LETSN	DSZ	MSFLG	ENABLE ASSIGNMENT
2699	04275	160202		JSM	FSCA,I	GET FORMULA
2700	04276	025714		LDB	SFLAG	
2701	04277	014133		CPB	M2	DID A STORE OCCUR
2702	04300	170402		RET		YES
2703	04301	064357		JMP	RETN2	NO

PAGE 57

9830 — MONITOR

```

2704*
2705* WRITE AND PRINT STATEMENT SYNTAX
2706*
2707 04302 160265 RITE JSM IOHEA,I GET DEVICE AND FORMAT
2708 04303 060534 JSM RPCKE GET RIGHT PAREN
2709 04304 055400 DSZ BADDR
2710 04305 055706 DSZ SBPTR
2711*
2712 04306 062315 PRINS JSM PRNTS CHECK SYNTAX (TO ACTST IF OK)
2713 04307 160206 SYE15 JSM ERR,I IMPROPER SYNTAX
2714*
2715* PRINT STATEMENT SYNTAX SUBROUTINE
2716*
2717 04310 010043 PRIN1 CPA B42 QUOTE?
2718 04311 066325 JMP PRIN2+4 YES
2719 04312 060461 JSM SYMC2
2720 04313 012001 DEF COMMA-1 COMMA OR SEMICOLON?
2721 04314 066321 JMP PRIN2 NO
2722 04315 024132 PRNTS LDB M1
2723 04316 035712 STB PFLAG ENABLE FORMULA AND TAB
2724 04317 061066 JSM GETSK
2725 04320 065150 JMP EOST-1
2726 04321 010043 PRIN2 CPA B42 QUOTE?
2727 04322 066324 JMP *+2 YES
2728 04323 066334 JMP PRIN3 NO
2729 04324 045706 ISZ SBPTR
2730 04325 160245 JSM GTSTA,I RECORD STRING CONSTANT
2731 04326 061153 JSM EOLCK EOL?
2732 04327 074742 SBR 16
2733 04330 135706 STB SBPTR,I CLEAR OUT NEXT WORD
2734 04331 024132 LDB M1
2735 04332 035712 STB PFLAG ENABLE FORMULA
2736 04333 066310 JMP PRIN1
2737 04334 045712 PRIN3 ISZ PFLAG TAB OR FORMULA PERMITTED?
2738 04335 170402 RET NO, RETURN, IMPROPER SYNTAX
2739 04336 026354 LDB TABA
2740 04337 160175 JSM TSRHA,I LOOK FOR "TAB"
2741 04340 066350 JMP PRIN4 NO
2742 04341 070544 SAL 5
2743 04342 040067 IOR TYPFL SET PRE-DEFINED FUNCTION FLAGS
2744 04343 060454 JSM SBPUD-2 STORE CODE FOR TAB
2745 04344 024123 LDB RBOP
2746 04345 135706 STB SBPTR,I
2747 04346 160202 JSM FSCA,I GET TAB FORMULA
2748 04347 066352 JMP PRIN5
2749 04350 055400 PRIN4 DSZ BADDR BACK UP POINTER
2750 04351 160202 JSM FSCA,I FETCH FORMULA
2751 04352 061153 PRIN5 JSM EOLCK EOL?
2752 04353 066310 JMP PRIN1 NO
2753 04354 023773 TABA DEF TAB+TAB+1

```

PAGE 58

9830 — MONITOR

2755*

2756*

OUTPUT ERROR # GIVEN IN P+1

2757*

2758

04355 060405

AERR

JSM XFARI

SAVE AR2 IN AR1

2759

04356 060400

JSM GPARM

GET ERROR # IN (B)

2760

04357 066370

JMP ERR1

2761*

2762*

OUTPUT ERROR # FROM ERROR TABLE

2763*

2764

04360 055777

ERROR

DSZ RSPTR

ERROR SOURCE IN (A)

2765

04361 121777

LDA RSPTR,I

TABLE ADDRESS IN (B)

2766

04362 026476

LDB ERBS

INC ADDRESS

2767

04363 074070

SIB *+1

MATCH?

2768

04364 074457

CPA B,I

YES

2769

04365 074076

TCB

CONTINUE IF NO MATCH

2770

04366 075653

SBP *-3

COMPUTE NEGATIVE OF ERROR #

2771

04367 006476

ADB ERBS

SAVE —(ERROR #)

2772

04370 035706

ERR1

STB SBPTR

2773

04371 021450

LDA PMODE

2774

04372 072350

RZA ERR2

2775

04373 021433

LDA TFLAG

2776

04374 070250

SZA ERR2

SKIP IF NOT PTAPE

2777

04375 020060

LDA .100

2778

04376 024211

LDB .BUFA

PRINT LINE IF PTAPE AND P—ALL

2779

04377 061337

JSM NDWRT

2780

04400 061344

JSM CRLF1

2781

04401 021416

LDA MODE

2782

04402 070112

SAM *+2

SKIP IF SYNTAX MODE

2783

04403 160264

JSM CRCKA,I

TERMINATE CURRENT LINE

2784

04404 160170

JSM CLERA,I

CLEAR BUFFER

2785

04405 035622

STB TERM

SET UNTERMINATED FLAG

2786

04406 020127

LDA B170K

2787

04407 031447

STA SLCOD

SET TO STANDARD PRINTER

2788

04410 020023

LDA .6

2789

04411 026464

LDB EMSGA

PUT "ERROR" IN BUFFER

2790

04412 061335

JSM NPWRT

2791

04413 025706

LDB SBPTR

2792

04414 074076

TCB

OUTPUT POSITIVE ERROR #

2793

04415 160210

JSM OUTIA,I

IS A LINE NUMBER

2794

04416 025427

LDB .LNUM

APPLICABLE

2795

04417 074310

SZB ERR3

2796

04420 020026

LDA .9

2797

04421 026465

LDB LMSGA

PUT "IN LINE" IN BUFFER

2798

04422 061335

JSM NPWRT

2799

04423 025427

LDB .LNUM

2800

04424 160210

JSM OUTIA,I

OUTPUT LINE #

2801

04425 045400

ERR3

ISZ BADDR

OUTPUT 2 BLANKS

2802

04426 045400

ISZ BADDR

2803

04427 021433

LDA TFLAG

2804

04430 072110

RZA *+2

2805

04431 072053

SAP *+1,S

SET TO "PRINT" IF PTAPE

2806

04432 031453

STA PRFLG

SET "DISP" FLAG

2807

04433 160176

JSM ELINA,I

OUTPUT ERROR

2808

04434 160264

JSM CRCKA,I

GIVE CRLF IF P—ALL OR PTAPE

2809

04435 025706

ERR4

LDB SBPTR

RECALL —(ERROR #)

2810

04436 014132

CPB M1

ERROR !?

180

168

1,444,141

168

PAGE 59

9830 — MONITOR

2811	04437	066445		JMP	ERN1	YES
2812	04440	004063		ADB	.99	
2813	04441	074313		SBP	ERR5	SKIP IF NON-RECOVERABLE
2814	04442	035442		STB	STPFL	SET TO STOP AT END OF LINE
2815	04443	035622		STB	TERM	SET UNTERMINATED LINE FLAG
2816	04444	064444		JMP	XFAR2	RESTORE AR2 AND RETURN
2817*						
2818	04445	035421	ERN1	STB	PRADD	SET TO ABORT DECOMPILE
2819	04446	035416		STB	MODE	SET TO SCRATCH SYMTAB
2820	04447	021432	ERR5	LDA	DRQST	
2821	04450	070110		SZA	*+2	IN "INPUT" STATEMENT?
2822	04451	165432		JMP	DRQST,I	YES, RETURN
2823	04452	021416		LDA	MODE	FIND OUT WHERE ERROR OCCURRED
2824	04453	010132		CPA	M1	MFASE?
2825	04454	066460		JMP	ERET-2	YES
2826	04455	070210		SZA	ERET-1	SKIP IF PROGRAM EXECUTION
2827	04456	070212		SAM	ERET	SKIP IF SYNTAX PHASE
2828	04457	166477		JMP	CALXA,I	MUST HAVE BEEN CALC EXECUTION
2829	04460	060645		JSM	MINIT	SCRATCH INCOMPLETE SYMTAB
2830	04461	060653		JSM	INITS+2	SCRATCH STACKS AND RESET
						PROGRAM POINTER
2831	04462	045701	ERET	ISZ	NDISP	SET TO NOT DISPLAY AUTO #
2832	04463	066047		JMP	PEXM2	
2833	04464	011154	EMSGA	DEF	ERRM+ERRM	POINTER TO "ERROR"
2834	04465	011162	LMSGGA	DEF	ERRM+ERRM+6	POINTER TO "IN LINE"
2835	04466	042522	ERRM	ASC	8,ERROR	IN LINE
	04467	051117				
	04470	051040				
	04471	020111				
	04472	047040				
	04473	046111				
	04474	047105				
	04475	020040				
2836	04476	000267	ERBS	DEF	E1-1	
2837	04360		MERR	EQU	ERROR	MFASE ERROR
2838	04477	010070	CALXA	DEF	CALEX	

PAGE 60 9830—UPDATE PROGRAM

2840* DELETE STATEMENT FROM PROGRAM

2841*					
2842	04500	160242	DELST	JSM DCOMA,I	DECOMPILE
2843	04501	061072		JSM FNDPS-3	FIND STATEMENT TO BE DELETED
2844	04502	066506		JMP EDITR	DOESNT EXIST
2845	04503	066506		JMP EDITR	
2846	04504	070110		SZA EDITR	SKIP IF TYPING AID KEY
2847	04505	062571		JSM CLPRG	CLOSE UP PROGRAM
2848	04506	060651	EDITR	JSM INITS	INITIALIZE STACK POINTERS
2849	04507	062723		JSM INCLN	INCREMENT LINE NUMBER
2850	04510	066042		JMP PEXMK-2	

2851* ACCEPT STATEMENT

2852*					
2853*					
2854	04511	020112	ACTST	LDA RSTKA	
2855	04512	031777		STA RSPTR	RESET RETURN STACK
2856	04513	021416		LDA MODE	
2857	04514	070073		SAP *+1,C	SWITCH OUT OF
2858	04515	031416		STA MODE	SYNTAX MODE
2859	04516	070110		SZA *+2	
2860	04517	164247		JMP ECALA,I	JUMP TO CALC EXECUTION
2861	04520	160242	ACTKY	JSM DCOMA,I	DECOMPILE
2862	04521	020233		LDA SBUFA	
2863	04522	070076		TCA	COMPUTE LENGTH
2864	04523	001706		ADA SBPTR	OF STATEMENT
2865	04524	031473		STA TEMP	SAVE LENGTH IN TEMP
2866	04525	025551		LDB RBUFF	READ LINE #
2867	04526	074450		SZB ACCS1	SKIP IF TYPING AID DEFINITION
2868	04527	041552		IOR RBUFF+1	MERGE INTO OP-CODE WORD
2869	04530	031552		STA RBUFF+1	
2870	04531	061072		JSM FNDPS-3	FIND ADDRESS IN PROGRAM
2871	04532	066565		JMP ACCS4	APPEND
2872	04533	066565		JMP ACCS4	INSERT
2873	04534	070150		SZA ACCS1	SKIP IF REPLACING TYPING AID KEY
2874	04535	061007		JSM NEXTA	NO, COMPUTE LENGTH AND END OF STATEMENT

2875	04536	066544		JMP ACCS3	
2876	04537	021424	ACCS1	LDA PBUFF	SET POINTERS TO
2877	04540	031476		STA TEMP3	REPLACE
2878	04541	070076		TCA	
2879	04542	001423		ADA PBPTR	ENTIRE PROGRAM
2880	04543	025423		LDB PBPTR	
2881	04544	070076	ACCS3	TCA	COMPARE
2882	04545	001473		ADA TEMP	LENGTHS
2883	04546	070250		SZA ACCS2	EQUAL
2884	04547	070112		SAM *+2	
2885	04550	066566		JMP ACCS4+1	OLD LINE IS SHORTER
2886	04551	021473		LDA TEMP	LONGER
2887	04552	062573		JSM CLPRG+2	CLOSE UP PROGRAM
2888*					
2889	04553	024233	ACCS2	LDB SBUFA	INITIALIZE POINTER
2890	04554	074517		LDA B,I	
2891	04555	131476		STA TEMP3,I	WORD
2892	04556	045476		ISZ TEMP3	
2893	04557	055473		DSZ TEMP	FINISHED?
2894	04560	077630		RIB ACCS2+1	NO
2895	04561	121424		LDA PBUFF,I	YES



PAGE 61

9830—UPDATE PROGRAM

2896	04562	072110	RZA	*+2	0 LINE NUMBER?
2897	04563	061225	JSM	LDEF	YES, END KEY DEFINITION
2898	04564	066506	JMP	EDITR	NO
2899*	INSERT STATEMENT				
2900	04565	021473	ACCS4	LDA	TEMP
2901	04566	062625	JSM	OVCHK	CHECK FOR OVERFLOW
2902	04567	061121	JSM	MVTOH	MOVE PROGRAM UP
2903	04570	066553	JMP	ACCS2	
2904*					
2905*	CLOSE UP PROGRAM				
2906*	ADDRESS OF LINE BEING CHANGED IS IN TEMP3				
2907*					
2908	04571	061007	CLPRG	JSM	NEXTA
2909	04572	070742	SAR	16	GET ADDRESS OF FIRST WORD MOVED
2910	04573	001476	ADA	TEMP3	SKIP (A) LOCATIONS FORM TEMP3
2911	04574	031477	STA	TEMP4	AND SAVE DESTINATION ADDRESS
2912*	TEMP4= DESTINATION ADDR OF FIRST WORD				
2913*	PBPTO= SOURCE ADDRESS OF LAST WORD +1				
2914*	(B)= SOURCE ADDRESS OF FIRST WORD				
2915	04575	015426	CLPR1	CPB	PBPTO
2916	04576	066603	JMP	CLPR2	ALL OF PROGRAM MOVED?
2917	04577	074517	LDA	B,I	YES
2918	04600	131477	STA	TEMP4,I	MOVE WORD FROM SOURCE
2919	04601	045477	ISZ	TEMP4	TO DESTINATION ADDRESS
2920	04602	077570	RIB	CLPR1	INC DESTINATION AND
2921	04603	021477	LDA	TEMP4	SOURCE ADDRESSES THEN JUMP
2922	04604	074076	TCB		RECALL NEW PBPTO
2923	04605	070037	ADB	A	COMPUTE NET CHANGE
2924	04606	031426	STA	PBPTO	IN PROGRAM LENGTH
2925	04607	021423	LDA	PBPTR	STORE NEW END OF MAINLINE
2926	04610	074017	ADA	B	
2927	04611	031423	STA	PBPTR	UPDATE END OF CURRENT PROGRAM
2928	04612	021422	LDA	UKPTR	GET KEY DEFINITION FLAG
2929	04613	070450	SZA	OVCHK-1	RETURN IF NOT IN KEY DEFINITION
2930	04614	070517	LDA	A,I	
2931	04615	074017	ADA	B	
2932	04616	131422	STA	UKPTR,I	UPDATE KEY LENGTH
2933	04617	021425	LDA	PBUFO	
2934	04620	074017	ADA	B	
2935	04621	031425	STA	PBUFO	UPDATE MAINLINE PBUFF
2936	04622	005405	ADB	FWUP	
2937	04623	035405	STB	FWUP	UPDATE END OF KEY AREA POINTER
2938	04624	170402	RET		
2939*					
2940*	CHECK FOR PROGRAM SPACE OVERFLOW				
2941*	(A)= NEW WORD REQUIREMENT				
2942	04625	025426	OVCHK	LDB	PBPTO
2943	04626	035475	STB	TEMP2	SET SOURCE ADDRESS
2944	04627	031711	STA	DFLAG	FRO PROGRAM RELOCATION
2945	04630	070037	ADB	A	SAVE COUNT
2946	04631	035477	STB	TEMP4	SET DESTINATION
2947	04632	001426	ADA	PBPTO	ADDRESS
2948	04633	074076	TCB		COMPUTE NEW END OF MAINLINE
2949	04634	005407	ADB	SYMTP	ENOUGH SPACE FOR
2950	04635	074113	SBP	*+2	PROGRAM
2951	04636	064561	JMP	FSCE4	BUFFER: SKIP IF YES
					NO, SYNTAX OVERFLOW

PAGE 62

9830—UPDATE PROGRAM

2952	04637	025711	LDB	DFLAG	RECALL COUNT
2953	04640	066606	JMP	CLPR2+3	

PAGE 63

9830 — SYSTEM COMMANDS

2955*

2956*

SCRATCH SYSTEM COMMAND

2957*

2958

04641

061055

SCRAT

JSM

GNEXT

GET NEXT CHAR

2959

04642

000154

ADA

M64

2960

04643

010012

CPA

.1

"A"?

2961

04644

066000

JMP

ENTRY

YES, SCRATCH ALL

2962

04645

010027

CPA

.11

"K"?

2963

04646

066652

JMP

SCRK

YES, SCRATCH KEYS

2964

04647

010035

CPA

.22

"V"?

2965

04650

066035

JMP

KILLV

YES, SCRATCH VARIABLES

2966

04651

066031

JMP

KILLP

OTHERWISE SCRATCH PROGRAM

2967

04652

061225

SCRK

JSM

LDEF

INSURE NOT IN KEY MODE

2968

04653

063711

JSM

DCOMP

2969

04654

020166

LDA

FWAM

2970

04655

025405

LDB

FWUP

2971

04656

164254

JMP

XSCRA,I

SCRATCH ALL KEYS

2972*

2973*

DELETE SYSTEM COMMAND

2974*

2975

04657

060730

DELET

JSM

GETAD

GET ADDRESSES

2976

04660

035476

STB

TEMP3

SAVE SOURCE ADDRESS

2977

04661

160242

JSM

DCOMA,I

DECOMPILE

2978

04662

025476

LDB

TEMP3

2979

04663

062575

JSM

CLPR1

CLOSE UP PROGRAM

2980

04664

066036

JMP

RDYPT-1

RE INITIALIZE

2981*

2982*

RENUMBER SYSTEM COMMAND

2983*

2984

04665

162733

RENUM

JSM

GETLA,I

GET OPTIONAL PARAMETERS

2985

04666

020061

LDA

.10

2986

04667

024061

LDB

.10

2987

04670

013470

STA

L2T1

SAVE STARTING LINE NUMBER

2988

04671

035471

STB

L2T2

SAVE INCREMENT

2989

04672

063675

JSM

COMPL

CONVERT OT ADDRESS FORMAT

2990

04673

025471

LDB

L2T2

2991

04674

035430

STB

LNINC

SET LINE INCREMENT

2992

04675

024133

LDB

M2

SET FLAG

2993

04676

035471

STB

L2T2

FOR 1ST PASS

2994

04677

021470

RENMI

LDA

L2T1

SET STARTING

2995

04700

031427

STA

.LNUM

LINE NUMBER

2996

04701

025424

LDB

PBUF

INITIALIZE PROGRAM

2997

04702

035472

RENMI

STB

TEMPS

POINTER

2998

04703

015423

CPB

PBPTR

END OF PROGRAM

2999

04704

066720

JMP

RENMI

YES

3000

04705

121472

LDA

TEMPS,I

3001

04706

070610

SZA

RENMI+2

SKIP IF TYPING AID

3002

04707

050062

AND

FLGBT

GET SECURITY BIT

3003

04710

041427

IOR

.LNUM

MERGE IN NEW LINE #.

3004

04711

024132

LDB

M1

3005

04712

015471

CPB

L2T2

SECOND PASS?

3006

04713

131472

STA

TEMPS,I

YES, STORE NEW LINE #.

3007

04714

025472

RENMI

LDB

TEMPS

COMPUTE ADDRESS

3008

04715

061007

JSM

NEXTA

OF NEXT LINE

3009

04716

062723

JSM

INCLN

NO, INCREMENT .LNUM

3010

04717

066702

JMP

RENMI

184

172

1,444,141

172

PAGE 64 9830 — SYSTEM COMMANDS

3011	04720	045471	RENMA	ISZ	L2T2	FIRST PASS?
3012	04721	066677		JMP	RENMI	YES, NOW DO SECOND PASS
3013	04722	066036		JMP	RDYPT-1	DECOMPILE AND RETURN
3014*						
3015*						
3016*						
3017	04723	021430	INCLN	LDA	LNINC	
3018	04724	001427		ADA	LNUM	
3019	04725	031427		STA	LNUM	
3020	04726	000161		ADA	MAXSN	UPDATE LINE NUMBER
3021	04727	070452		SAM	INCLR	GREATER THAN 9999?
3022	04730	070742		SAR	16	RETURN IF LESS THAN 9999
3023	04731	031430		STA	LNINC	
3024	04732	064726		JMP	SYE25	SET TO MANUAL
3025	04733	003647	GETLA	DEF	GETLN	JUMP TO IMPROPER INTEGER ERROR
3026*						
3027*						
3028*						
3029	04734	025405	SCRIP	LDB	FWUP	
3030	04735	035424		STB	PBUFF	
3031	04736	035423		STB	PBPTR	
3032	04737	035426		STB	PBPTO	
3033	04740	170402	INCLR	RET		

PAGE 65

9830 — PRE-EXECUTION PROCESSING

3035*
 3036* RUN SYSTEM COMMAND
 3037* (A)=STARTING LINE NUMBER
 3038*
 3039 04741 031725 RUN STA NXTST SAVE STARTING LINE NUMBER
 3040 04742 060645 JSM MINIT SCRATCH OLD SYMTAB
 3041 04743 070742 SAR 16
 3042 04744 031455 STA FORMT CLEAR THE FORMAT FLAG
 3043 04745 031432 STA DRQST
 3044 04746 020012 LDA .1
 3045 04747 031434 STA UNITS SET TO RADIANS
 3046 04750 021424 LDA PBUFF
 3047 04751 011423 CPA PBPTR NULL PROGRAM?
 3048 04752 164174 JMP RDYDA,1 YES
 3049 04753 125424 LDB PBUFF,1
 3050 04754 075710 SZB *-2 TYPING AID KEY?
 3051 04755 031421 STA PRADD INITIALIZE PROGRAM POINTER
 3052 04756 021405 LDA FWUP
 3053 04757 031730 STA COML INITIALIZE COMMON POINTER
 3054 04760 045412 ISZ LSTPT SET L-STACK POINTER = PBPTR
 3055 04761 020132 LDA M1 FLAG ERROR ROUTINE
 3056 04762 031416 STA MODE TO SCRATCH SYMTAB
 3057*
 3058* SKIP TO BEGINNING OF NEXT STATEMENT
 3059*
 3060 04763 025421 MSKIP LDB PRADD
 3061 04764 021407 LDA SYMTP GET POINTER FOR MILOP
 3062 04765 015423 CPB PBPTR END OF PROGRAM?
 3063 04766 067122 JMP MILOP YES
 3064 04767 063601 JSM NXTOP FETCH 1ST OPERAND IN NEXT STATEMENT
 3065 04770 066763 JMP MSKIP END OF STATEMENT
 3066 04771 067106 JMP MLOP2 INTEGER, SKIP IT
 3067 04772 031473 MSTND STA MBOX1 VARIABLE OPERAND, SAVE IT
 3068 04773 024132 LDB M1
 3069 04774 035474 STB MBOX1+1 INITIALIZE DIMENSIONS TO UNDEFINED
 3070 04775 063234 JSM FNTYP FIND OPERAND TYPE
 3071 04776 067106 JMP MLOP2 FUNCTION
 3072 04777 020132 LDA M1 STRING
 3073 05000 025731 MLOP7 LDB TYPE
 3074 05001 014020 CPB MDMA DIM?
 3075 05002 074742 MLP72 SBR 16 YES, SET (B)=0
 3076 05003 014017 CPB MCOMA COM?
 3077 05004 067112 JMP MLP71
 3078 05005 070153 SAP *+3 SKIP IF SIMPLE VARIABLE
 3079 05006 076150 RZB MLOP5 ARRAY, SKIP IF NOT DIM OR COM
 3080 05007 067017 JMP MLOP8
 3081 05010 076310 RZB ML2L SKIP IF NOT DIM OR COM
 3082*
 3083* PROCESS VARIABLE
 3084 05011 063306 MLOP5 JSM SSYMT+1 SEARCH SYMBOL TABLE
 3085 05012 067043 JMP MLOP9 NOT FOUND
 3086 05013 021731 LDA TYPE FOUND
 3087 05014 010017 CPA MCOMA COM STMT?
 3088 05015 067037 JMP MER5 YES, DOUBLE DEFINITION
 3089 05016 067106 ML2L JMP MLOP2 NO, PROCESS NEXT WORD
 3090* COM OR DIM

PAGE 66

9830—PRE-EXECUTION PROCESSING

3091* (A)—1 FOR 1 DIM OR STRING, -2 FOR 2 DIM

3092 05017 045472 MLOP8 ISZ TEMPS

3093 05020 045472 ISZ TEMPS

3094 05021 125472 LDB TEMPS,I READ FIRST DIMENSION

3095 05022 004132 ADB M1 OFFSET BY 1

3096 05023 074404 SBL 8 PLACE IT IN LEFT HALF WORD

3097 05024 010132 CPA M1 2 DIMENSIONS

3098 05025 067032 JMP *+5 NO, SET SECOND DIM TO 1

3099 05026 045472 ISZ TEMPS

3100 05027 045472 ISZ TEMPS

3101 05030 105472 ADB TEMPS,I ADD SECOND DIMENSION INTO RIGHT HALF

3102 05031 004132 ADB M1 OFFSET SECOND DIM

3103 05032 035474 STB MBOX1+ STORE PACKED DIMENSIONS

3104 05033 063306 JSM SSYMT+1 SEARCH SYMBOL TABLE

3105 05034 067043 JMP MLOP9 NOT FOUND

3106 05035 004132 ADB M1 FOUND, (B)=ADDRESS

3107 05036 074637 ISZ B,I ALREADY DIMENSIONED?

3108 05037 062360 MERR JSM MERR YES

3109 05040 021474 LDA MBOX1+1 STORE DIMENSIONS

3110 05041 074557 STA B,I

3111 05042 067106 JMP MLOP2

3112* ENTER NEW SYMBOL INTO TABLE

3113 05043 063404 MLOP9 JSM ESYMT ENTER INTO SYMBOL TABLE

3114 05044 021731 LDA TYPE

3115 05045 010017 CPA MCOMA COM STMT?

3116 05046 067050 JMP MCOM1 YES

3117 05047 067106 JMP MLOP2 NO

3118* ALLOCATE SPACE FOR COMMON VARIABLE

3119 05050 021473 MCOM1 LDA MBOX1 SET SIGN BIT IN

3120 05051 072053 SAP *+1,S SYMBOL TABLE TO

3121 05052 074557 STA B,I INDICATE COMMON

3122 05053 025474 LDB MBOX1+1 RECALL DIMENSIONS

3123 05054 063257 JSM MDIM COMPUTE NUMBER OF WORDS NEEDED

3124 05055 025730 LDB COML GET OLD COMMON POINTER

3125 05056 074017 ADA B

3126 05057 031730 STA COML UPDATE COMMON POINTER

3127 05060 021424 LDA PBUFF

3128 05061 045422 ISZ UKPTR

3129 05062 055422 DSZ UKPTR KEY MODE?

3130 05063 021425 LDA PBUFO YES, GET FIRST WORD OF MAINLINE

3131 05064 070056 CMA MAXIMUM COMMON EXCEEDED?

3132 05065 001730 ADA COML

3133 05066 070112 SAM *+2

3134 05067 062360 CMERR JSM MERR YES, TOO MUCH COMMON IN KEY

3135 05070 021474 LDA MBOX1+1 READ DIMENSIONS

3136 05071 010132 CPA M1 SIMPLE VARIABLE?

3137 05072 067103 JMP MCOM2 YES

3138 05073 064557 STA B,I STORE FORMAL DIMENSIONS

3139 05074 074070 SIB *+1

3140 05075 021473 LDA MBOX1

3141 05076 070206 RAR 5 IF BIT 4=1, IT MUST

3142 05077 070212 SAM MCOM2 BE A STRING VARIABLE

3143 05100 021474 LDA MBOX1+1 NO, MUST BE AN ARRAY

3144 05101 074557 STA B,I STORE ACTUAL DIMENSIONS

3145 05102 074070 SIB *+1 POINT TO FIRST VARIABLE OF ARRAY

3146 05103 021407 MCOM2 LDA SYMTP

PAGE 67

9830—PRE-EXECUTION PROCESSING

3147	05104	070070		SIA	*+1	
3148	05105	070577		SIB	A,I	STORE ADDRESS IN SYMTAB
3149	05106	063620	MLOP2	JSM	WXTPI	GET NEXT OPERAND OF STMT
3150	05107	066763		JMP	MSKIP	END OF STMT
3151	05110	067106		JMP	MLOP2	INTEGER, SKIP IT
3152	05111	066772		JMP	MSTND	VARIABLE OPERAND
3153*	CHECK FOR LEGAL COM					
3154	05112	125424	MLP71	LDB	PBUFF,I	
3155	05113	074073		SBP	*+1,C	
3156	05114	015427		CPB	LNUM	IS THIS THE FIRST LINE?
3157	05115	067002		JMP	MLP72	YES
3158	05116	164330		JMP	E30+3,I	NO, MISPLACED COM
3159*						
3160*	MFASE PART TWO					
3161*						
3162*	MOVE SYMTAB TO LOW MEMORY					
3163	05117	070537	M5LOP	LDB	A,I	MOVE SYMBOL
3164	05120	135412		STB	LSTPT,I	TABLE WORD
3165	05121	045412		ISZ	LSTPT	INC DESTINATION POINTER
3166	05122	011406	M1LOP	CPA	VALUE	ALL OF TABLE MOVED?
3167	05123	067125		JMP	*+2	YES
3168	05124	073570		RIA	M5LOP	INC SOURCE POINTER
3169	05125	070742		SAR	16	
3170	05126	031427		STA	LNUM	CLEAR LINE #
3171	05127	025412		LDB	LSTPT	
3172	05130	020062		LDA	FLGBT	SET ALL PROSPECTIVE
3173	05131	074557		STA	B,I	
3174	05132	015406		CPB	VALUE	VALUE TABLE WORDS
3175	05133	067135		JMP	*+2	
3176	05134	077670		RIB	*-3	TO UNDEFINED VALUE
3177*	DURING THIS SECTION USE LSTAK AS SYMTAB POINTER					
3178*	AND HSTPT AS VALUE TABLE POINTER					
3179	05135	035413		STB	HSTPT	INITIALIZE VALUE TABLE POINTER
3180	05136	021412		LDA	LSTPT	
3181	05137	031410		STA	LSTAK	INITIALIZE SYMBOL TABLE POINTER
3182*						
3183	05140	025410	M2LOP	LDB	LSTAK	
3184	05141	015426		CPB	PBPTO	MORE SYMBOLS?
3185	05142	067216		JMP	M4LOP	NO
3186	05143	055410		DSZ	LSTAK	YES, MOVE TO NEXT SYMBOL
3187	05144	121410		LDA	LSTAK,I	READ SYMBOL
3188	05145	055410		DSZ	LSTAK	
3189	05146	070233		SAP	*+4,C	DEFINED IN COM?
3190	05147	004132		ADB	M1	
3191	05150	074557		STA	B,I	YES, CLEAR FLAG
3192	05151	067140		JMP	M2LOP	BYPASS ALLOCATION PHASE
3193	05152	031473		STA	MBOX1	NO, SAVE SYMBOL
3194	05153	063234		JSM	FNTYP	FIND VARIABLE TYPE
3195	05154	067140		JMP	M2LOP	FUNCTION
3196	05155	067160		JMP	*+3	STRING
3197	05156	010134		CPA	M3	ARRAY OF UNKNOWN DIMENSIONS?
3198	05157	062360	MER10	JSM	MERR	YES
3199	05160	031474		STA	MBOX1+1	SAVE TYPE FLAG
3200	06161	125410		LDB	LSTAK,I	READ DIMENSIONS
3201	05162	014132		CPB	M1	DIMENSIONED?
3202	05163	067165		JMP	*+2	NO

PAGE 68

9830 — PRE-EXECUTION PROCESSING

3203	05164	067173	JMP	M3LOP	YES
3204	05165	074742	SBR	161	LOAD STD DIM FOR STRING
3205	05166	070213	SAP	M3LOP-1	SKIP IF SIMPLE VARIABLE OR STRING
3206	05167	027233	LDB	84400	LOAD STD DIMENSION FOR 1 DIM
3207	05170	010133	CPA	M2	2 DIMENSIONS?
3208	05171	004026	ADB	9	YES, CORRECT FOR 2 DIMS
3209	05172	135410	STB	LSTAK,I	
3210	05173	021473	M3LOP	LDA	MBOX1
3211	05174	063257	JSM	MDIM	RECALL SYMBOL
3212	05175	070076	TCA		COMPUTE STORAGE REQUIRED
3213	05176	025413	LDB	HSTPT	MOVE VALUE TABLE POINTER
3214	05177	063556	JSM	MOVCK+1	AND CHECK FOR OVERFLOW
3215	05200	021474	LDA	MBOX1+1	RECALL TYPE FLAG
3216	05201	025413	LDB	HSTPT	GET NEW VALUE TABLE POINTER
3217	05202	074070	SIB	*+1	
3218	05203	010076	CPA	B36	STRING?
3219	05204	067211	JMP	M6LOP	YES
3220	05205	070353	SAP	M6LOP+3	SKIP IF SIMPLE VARIABLE
3221	05206	121410	LDA	LSTAK,I	GET DIMENSIONS AND STORE
3222	05207	074557	STA	B,I	IN VALUE TABLE
3223	05210	074130	SIB	*+2	INC POINTER
3224	05211	121410	M6LOP	LDA	LSTAK,I
3225	05212	074557	STA	B,I	GET STRING DIMENSIONS
3226	05213	074070	SIB	*+1	STORE DIMENSION
3227	05214	135410	STB	LSTAK,I	IN VALUE TABLE
3228	05215	067140	JMP	M2LOP	STORE VALUE ADDRESS IN SYMTAB
3229*	CLEAN UP FOR EXECUTION				
3230	05216	021413	M4LOP	LDA	HSTPT
3231	05217	031406	STA	VALUE	SET VALUE TABLE POINTER
3232	05220	025412	LDB	LSTPT	MOVE SYMBOL
3233	05221	015426	CPB	PBPTO	
3234	05222	067230	JMP	*+6	
3235	05223	004132	ADB	M1	
3236	05224	074517	LDA	B,I	TABLE BACK
3237	05225	131413	STA	HSTPT,I	
3238	05226	055413	DSZ	HSTPT	TO HIGH
3239	05227	067221	JMP	M4LOP+3	MEMORY
3240	05230	021413	LDA	HSTPT	
3241	05231	031407	STA	SYMPT	SET SYMBOL TABLE POINTER
3242	05232	164004	JMP	XECA,I	MFASE COMPLETE
3243*					
3244	05233	004400	B4400	OCT	4400
3245*	DETERMINE THE TYPE OF AN OPERAND GIVEN IN (A)				
3246*					
3247*	RETURN	P+1:	FUNCTION		
3248*		P+2:	STRING		
3249*		P+3:	NUMERIC VARIABLE		
3250*			(A)=POSITIVE IF SIMPLE VARIABLE		
3251*			=-1 IF SINGLE DIM ARRAY		
3252*			=-2 IF DOUBLE DIM		
3253*			=-3 IF UNKNOWN DIM		
3254*					
3255*			(B)=4 IF FULL PRECISION		
3256*			=2 IF SPLIT PRECISION		
3257*			=1 IF INTEGER		
3258*					



3259	05234	050040	FNTYP	AND B37	ISOLATE TYPE
3260	05235	010076		CPA B36	STRING?
3261	05236	064357		JMP RETN2	YES
3262	05237	074742		SBR 16	
3263	05240	010040		CPA B37	FUNCTION?
3264	05241	170432		RET	YES
3265	05242	024021		LDB 4	SET TO FULL PRECISION
3266	05243	070146		RAR 4	
3267	05244	070171		SLA *+3,C	NUMBERED VARIABLE
3268	05245	070546		RAR 12	YES
3269	05246	064356		JMP RETN3	
3270	05247	070133		SAP *+2,C	INTEGER?
3271	05250	004134		ADB M3	YES
3272	05251	070646		RAR 14	
3273	05252	070131		SLA *+2,C	SPLIT PRECISION?
3274	05253	004133		ADB M2	YES
3275	05254	070646		RAR 14	RECALL NUMBER OF DIMENSIONS
3276	05255	070076		TCA	NEGATE
3277	05256	064356		JMP RETN3	

COMPUTE STORAGE NEEDED BY A VARIABLE

(A)=SYMBOL

(B)=PACKED DIMENSIONS FOR ARRAY OR STRING

STORAGE NEEDED: $STRING = (DIM1 + 1) / 2 + 1$
 $ARRAY = DIM1 * DIM2 * \# \text{ OF WORDS/ELEM} + 2$
 $SIMPLE \text{ VARIABLE} = \# \text{ OF WORDS}$

EXIT (A)=STORAGE NEEDED

3289	05257	035470	MDIM	STB L2T1	
3290	05260	063234		JSM FNTYP	
3291	05261	067277		JMP MDIM2	FUNCTION (B)=0
3292	05262	067301		JMP MDIM1	STRING
3293	05263	070613		SAP MDIM2	ARRAY?
3294	05264	021470		LDA L2T1	
3295	05265	050105		AND B377	MULTIPLY SECOND DIMENSION
3296	05266	070070		SIA *+1	BY THE NUMBER OF WORDS PER
3297	05267	061201		JSM IMPY	ELEMENT
3298	05270	025470		LDB L2T1	
3299	05271	074342		SBR 8	
3300	05272	074070		SIB *+1	
3301	05273	061201		JSM IMPY	MULTIPLY BY SECOND DIMENSION
3302	05274	000017		ADA 2	ADD WORDS FOR DIMENSIONS
3303	05275	070153		SAP MDIM2+1	GREATER THAN 32K?
3304	05276	062360		JSM MERR	YES, DIMENSIONS TOO LARGE
3305	05277	074117	MER9	LDA B	MOVE RESULT TO (A)
3306	05300	170402	MDIM2	RET	
3307	05301	074342		SBR 8	
3308	05302	004017	MDIM1	ADB 2	ADD 2 TO FORMAL DIMENSION
3309	05303	074002		SBR 1	DIVIDE BY 2
3310	05304	077570		RIB MDIM2	INCREMENT AND JUMP

PAGE 70

9830 — PRE-EXECUTION PROCESSING

3312* SEARCH SYMBOL TABLE
 3313* RETURN P+1 IF NOT FOUND
 3314* P+2 IF FOUND: ADDRESS IN B
 3315* IF BIT 4=1 THEN SYMBOL AND TYPE MUST MATCH EXACTLY
 3316* AN UNSPECIFIED PRECISION VARIABLE MATCHES A SPECIFIED PRECISION
 3317* AND THE SPECIFIED PRECISION IS INSERTED IN THE TABLE
 3318* AN ARRAY WITH UNSPECIFIED NUMBER OF DIMENSIONS MATCHES A
 3319* SPECIFIED ARRAY AND THE DIMENSION NUMBER IS PUT IN THE TABLE
 3320* AN INCONSISTENCY IN DIMENSIONS OR PRECISION CAUSES AN ERROR
 3321*
 3322*
 3323*
 3324*
 3325*
 3326*
 3327*
 3328 05305 031473 SSYMT STA MBOX1 (A)=SYMBOL TO BE FOUND
 3329 05306 021473 LDA MBOX1
 3330 05307 050020 AND .3 SAVE SIMPLE/ARRAY TYPE
 3331 05310 031466 STA STMP2
 3332 05311 021473 LDA MBOX1
 3333 05312 050030 AND .12 SAVE PRECISION TYPE
 3334 05313 031467 STA STMP3 RECALL VARIABLE
 3335 05314 021473 LDA MBOX1 GET MASK TO MASK OFF PRECISION
 3336 05315 024112 LDB B1760
 3337 05316 070146 RAR 4
 3338 05317 070111 SLA *+2 BIT 4=1?
 3339 05320 024115 LDB B1777 YES, DONT MASK OFF ANY BITS
 3340 05321 070546 RAR 12 RESTORE VARIABLE
 3341 05322 074257 AND B MASK
 3342 05323 035470 STB MASK STORE MASK
 3343 05324 031471 STA STMP1 STORE PART OF SYMBOL TO MATCH
 3344 05325 014115 CPB B1777 MUST MATCH WHOLE SYMBOL?
 3345 05326 067373 JMP SVALU-1 YES, SEARCH AND RETURN FROM SVALU
 3346 05327 063373 JSM SVALU-1 NO, SEARCH FOR NAME ONLY
 3347 05330 170402 RET NOT FOUND
 3348 05331 021466 SSYM1 LDA STMP2 SEARCHING FOR
 3349 05332 072350 RZA SSYM4 SIMPLE VARIABLE?
 3350 05333 074517 LDA B,I YES, SIMPLE
 3351 05334 050020 AND .3 VARIABLE
 3352 05335 070410 SZA SM2L FOUND?
 3353 05336 063402 SSYM6 JSM SVAL1 NO, CONTINUE SEARCH
 3354 05337 170402 RET
 3355 05340 067331 JMP SSYM1
 3356* SEARCHING FOR AN ARRAY
 3357*
 3358 05341 074517 SSYM4 LDA B,I
 3359 05342 050020 AND .3
 3360 05343 071550 SZA SSYM6 ARRAY FOUND? SKIP IF NO
 3361 05344 011466 CPA STMP2 DO DIMENSIONS MATCH?
 3362 05345 067361 JMP SSYM2 YES, NOW CHECK PRECISION
 3363 05346 010020 CPA .3 NO, TABLE VARIABLE UNSPECIFIED?
 3364 05347 067354 JMP SSYM5 YES
 3365 05350 021466 LDA STMP2 NO, SEARCH
 3366 05351 010020 CPA .3 VARIABLE UNSPECIFIED?
 3367 05352 067361 JMP SSYM2 YES, OK

PAGE 71

9830—PRE-EXECUTION PROCESSING

3368 05353 062360 SYME2 JSM MERR NO, INCONSISTENT DIMENSIONS
 3369 05354 074517 SSYM5 LDA B,I INSERT
 3370 05355 050135 AND M4
 3371 05356 041466 IOR STMP2 DIMENSION NUMBER
 3372 05357 074557 STA B,I
 3373 05360 067361 JMP SSYM2 INTO TABLE
 3374* PROPER TYPE FOUND, NOW CHECK FOR PROPER PRECISION
 3375*
 3376 05361 021467 SSYM2 LDA STMP3
 3377 05362 070350 SZA SSYM3-1 SEARCH VARIABLE UNSPECIFIED?
 3378 05363 074517 LDA B,I NO
 3379 05364 050030 AND .12
 3380 05365 072250 RZA SSYM3 TABLE VARIABLE UNSPECIFIED?
 3381 05366 021467 LDA STMP3 YES
 3382 05367 074617 IOR B,I
 3383 05370 074557 STA B,I INSERT SPECIFICATION IN TABLE
 3384 05371 064357 JMP RETN2 FOUND
 3385 05372 062360 SSYM3 JSM MERR PRECISION DOUBLY DEFINED
 3386*
 3387* SEARCH SYMBOL TABLE FOR A MATCH—(B),I AND MASK=STMP1
 3388*
 3389* RET P+1 IF NO MATCH FOUND
 3390* P+2 IF MATCH FOUND, (B)=ADDRESS
 3391 05373 025406 LDB VALUE
 3392 05374 015407 SVALU CPB SYMTP ALL TABLE SEARCHED?
 3393 05375 170402 RET YES, NO MATCH FOUND
 3394 05376 074517 LDA B,I NO, READ TABLE ENTRY
 3395 05377 051470 AND MASK GET THE BITS TO BE COMPARED
 3396 05420 011471 CPA STMP1 MATCH?
 3397 05401 064357 JMP RETN2 YES, RET P+2, (R)=ADDRESS
 3398 05402 004133 SVAL1 ADB M2 UPDATE TABLE POINTER
 3399 05403 067374 JMP SVALU
 3400 01470 MASK EQU L2T1
 3401 01471 STMP1 EQU L2T2
 3402 01466 STMP2 EQU LOCL1
 3403 01467 STMP3 EQU LOCL2
 3404*
 3405* ENTER 2 WORDS GIVEN IN MBOX1 AND MBOX1+1 INTO SYMBOL TABLE
 3406*
 3407 05404 020133 ESYM1 LDA M2 MOVE H-STOCK
 3408 05405 025407 LDB SYMTP
 3409 05406 063555 JSM MOVCK DOWN BY 2 WORDS
 3410 05407 074070 SIB *+1 ((B)=NEW SYMTP)
 3411 05410 021474 LDA MBOX1+1 TRANSFER WORDS INTO SYMTAB
 3412 05411 074557 ESYM1 STA B,I
 3413 05412 074070 SIB *+1
 3414 05413 021473 LDA MBOX1
 3415 05414 074557 STA B,I
 3416 05415 170402 RET
 3417*
 3418* ENTER A SIMPLE VARIABLE INTO SYMTAB AND VALUE TABLE
 3419*
 3420 05416 021473 ENTER LDA MBOX1
 3421 05417 063234 JSM FNTYP FIND OPERAND TYPE
 3422 05420 067437 JMP ENTE1 FUNCTION
 3423 05421 160206 NINIT JSM ERRA,I STRING, NOT INITIALIZED

PAGE 72

9830—PRE-EXECUTION PROCESSING

3424	05422	071752		SAM *-1	SIMPLE VARIABLE?
3425	05423	025406		LDB VALUE	YES
3426	05424	020137		LDA M6	NO, MOVE SYMTAB AND H-STACK
3427	05425	063555		JSM MOVCK	DOWN BY 6 WORDS
3428	05426	000135		ADA M4	COMPUTE ADDRESS OF VARIABLE
3429	05427	031474		STA MBOX1+1	
3430	05430	070137		LDB A	
3431	05431	020062		LDA FLGBT	
3432	05432	074557		STA B,I	SET TO UNDEFINED
3433	05433	004133		ADB M2	COMPUTE ADDRESS OF
3434	05434	063410		JSM ESYM+4	SYMTAB ENTRY AND ENTER
3435	05435	035406		STB VALUE	UPDATE VALUE TABLE POINTER
3436	05436	170402		RET	
3437*	SEARCH FOR A FUNCTION NOT FOUND IN SYMTAB				
3438*					
3439	05437	025406	ENTE1	LDB VALUE	OPEN UP 2 WORD SPACE
3440	05440	020133		LDA M2	
3441	05441	063555		JSM MOVCK	
3442	05442	025406		LDB VALUE	
3443	05443	004132		ADB M1	
3444	05444	035750		STB YC	SET UP ADDRESS POINTER
3445	05445	070742		SAR 16	INITIALIZE ADDRESS WORD TO 0
3446	05446	063411		JSM ESYM1	MAKE SYMTAB ENTRY
3447*					
3448*	SEARCH FOR FUNCTION WHICH HAS 0 ADDRESS				
3449	05447	063513	ENTFN	JSM SSTT	SAVE PROGRAM POINTERS
3450	05450	025424		LDB PBUFF	
3451	05451	063524		JSM SERFN	SEARCH CURRENT PROGRAM FIRST
3452	05452	024166		LDB FWAM	NOT FOUND
3453*	SEARCH FIRST LINE OF EACH KEY				
3454	05453	015405	ENTE3	CP8 FWUP	END OF KEY AREA?
3455	05454	067503		JMP ENTE5	YES, NOT FOUND
3456	05455	074070		SIB *+1	
3457	05456	074517		LDA B,I	
3458	05457	074070		SIB *+1	
3459	05460	010012		CPA .1	NULL KEY?
3460	05461	067453		JMP ENTE3	YES
3461	05462	035472		STB TEMPS	
3462	05463	121472		LDA TEMPS,I	
3463	05464	004132		ADB M1	
3464	05465	070610		SZA ENTE4+2	TYPING AID KEY?
3465	05466	060602		JSM NEXT1-4	NO, GET PAGE AND OP CODE
3466	05467	001466		ADA LOCL1	
3467	05470	013554		CPA DEFOP	DEF STMT?
3468	05471	067473		JMP *+2	YES
3469	05472	067477		JMP ENTE4	NO
3470	05473	121472		LDA TEMPS,I	
3471	05474	050115		AND B1777	
3472	05475	011473		CPA MBOX1	IS IT THE FUNCTION WE WANT?
3473	05476	067536		JMP SERF1	YES
3474	05477	025472	ENTE4	LDB TEMPS	NO
3475	05500	004134		ADB M3	
3476	05501	074437		ADB B,I	FIND NEXT KEY
3477	05502	067453		JMP ENTE3	
3478*	SEARCH MAINLINE LAST				
3479	05503	021422	ENTE5	LDA UKPTR	

PAGE 73

9830—PRE-EXECUTION PROCESSING

3480 05504 070250 SZA ENTE6 KEY MODE?
 3481 05505 025426 LDB PBPTO
 3482 05506 035423 STB PBPTR
 3483 05507 025425 LDB PBUFO YES, SERCH MAINLINE
 3484 05510 063524 JSM SERFN
 3485 05511 063543 ENTE6 JSM RSTT NOT FOUND, UNDEFINED
 3486 05512 164337 JMP E40,I
 3487*
 3488* SUBROUTINE TO SAVE PROGRAM POINTERS
 3489*
 3490 05513 025421 SSTT LDB PRADD
 3491 05514 035741 STB XC+1
 3492 05515 025472 LDB TEMPS
 3493 05516 035740 STB XC
 3494 05517 025427 LDB LNUM
 3495 05520 035742 STB XC+2
 3496 05521 025423 LDB PBPTR
 3497 05522 035743 STB XC+3
 3498 05523 170402 SERFR RET
 3499*
 3500* SUBROUTINE TO SEARCH A PROGRAM FOR A FUNCTION DEF
 3501*
 3502* (B)=ADDRESS OF FIRST WORD OF PROGRAM
 3503* RETURN IF NOT FOUND
 3504* RETURN TO SECOND RETURN ON STACK IF FOUND; STORE ADDRESS IN XC+3,I
 3505*
 3506 05524 035421 SERFN STB PRADD
 3507 05525 023554 LDA DEFOP
 3508 05526 160241 JSM SYTSA,I SEARCH FOR DEF STMT
 3509 05527 075613 SBP SERFR FOUND?
 3510 05530 121472 LDA TEMPS,I YES
 3511 05531 050115 AND B1777
 3512 05532 011473 CPA MBOX1 IS IT THE FN WE WANT?
 3513 05533 067535 JMP *+2 YES
 3514 05534 067525 JMP SERFN+1
 3515 05535 055777 DSZ RSPTR FOUND, DINK RETURN OFF STACK
 3516 05536 025472 SERFI LDB TEMPS
 3517 05537 004133 ADB M2
 3518 05540 135750 STB YC,I STORE ADDRESS IN SYMTAB
 3519 05541 025750 LDB YC
 3520 05542 074070 SIB *+1 GET SYMBOL TABLE ADDRESS
 3521*
 3522 05543 021740 RSTT LDA XC RESTORE PROGRAM POINTERS
 3523 05544 031472 STA TEMPS
 3524 05545 021741 LDA XC+1
 3525 05546 031421 STA PRADD
 3526 05547 021742 LDA XC+2
 3527 05550 031427 STA LNUM
 3528 05551 021743 LDA XC+3
 3529 05552 031423 STA PBPTR
 3530 05553 170402 RET
 3531*
 3532 05554 011774 DEFOP OCT 11774
 3533*
 3534* MOVE H-STACK DOWN AND CHECK FOR OVERFLOW
 3535*

184

182

1,444,141

182

PAGE 74

9830—PRE-EXECUTION PROCESSING

```

3536*      MOVE DOWN—(A) WORDS
3537*      (B)= ADDRESS OF LAST WORD MOVED
3538*
3539      05555 074070 MOVCK SIB *+1
3540      05556 031726 STA MBINI SAVE SPACE REQUIREMENT
3541      05557 035466 STB LOCL1 SAVE LAST WORD + 1
3542      05560 001413 ADA HSTPT COMPUTE NEW HSTPT
3543      05561 070137 LDB A
3544      05562 060555 JSM BHSTP+2 CHECK FOR OVERFLOW
3545      05563 070137 LDB A
3546      05564 021413 LDA HSTPT GET OLD HSTPT
3547      05565 035413 STB HSTPT STOKE NEW HSTPT
3548      05566 035727 STB MBIN2 STORE SOURCE POINTER
3549      05567 011466 MOVCI CPA LOCL1 ALL WORDS MOVED?
3550      05570 067575 JMP MOVCI2 YES
3551      05571 070537 LDB A,I MOVE
3552      05572 135727 STB MBIN2,I WORD
3553      05573 045727 ISZ MBIN2
3554      05574 073570 RIA MOVCI2 UPDATE POINTERS
3555      05575 025407 MOVCI2 LDB SYMTP
3556      05576 005726 ADB MBINI
3557      05577 035407 STB SYMTP UPDATE SYMBOL TABLE POINTER
3558      05600 170402 NXTPR RET
3559*      ROUTINE TO FETCH THE NEXT OPERAND FROM THE PROGRAM
3560*      TEMPS POINTS TO CURRENT WORD OF CODE; PRADD POINTS TO END OF LINE
3561*      CONSTANTS; STRING CONSTANTS; NON-RESEQUENCEABLE INTEGERS ARE
                                         SKIPPE
3562*      EXIT P+1: END OF LINE
3563*      P+2: RESEQUENCEABLE INTEGER FOUND
3564*      P+3: VARIABLE OPERAND FOUND
3565      05601 060572 NXTOP JSM NEXTL-2 COMPUTE END OF CURRENT LINE
3566      05602 005466 ADB LOCL1 ADD OP-CODE AND
3567      05603 007772 ADB MREM CONSTANT TO PAGE TO GET
3568      05604 035731 STB TYPE UNIQUE VALUE FOR STATEMENT
                                         TYPE
3569      05605 055472 DSZ TEMPS
3570      05606 121472 LDA TEMPS,I
3571      05607 050107 AND B1000 GET ASCII FLAG
3572      05610 073410 RZA NXTPR ASCII?
3573      05611 067620 JMP NXTP1
3574*      PROCESS A STRING CONSTANT
3575      05612 125472 NXTP2 LDB TEMPS,I COMPUTE
3576      05613 074070 SIB *+1
3577      05614 074404 SBL 8 LENGTH OF
3578      05615 074402 SBR 9 STRING
3579      05616 005472 ADB TEMPS SKIP TO
3580      05617 035472 STB TEMPS END OF CONSTANT
3581*      PROCESS NEXT WORD: RE-ENTER ROUTINE HERE AFTER
3582*      PROCESSING A VARIABLE OPERAND
3583      05620 045472 NXTP1 ISZ TEMPS SET TO NEXT WORD
3584      05621 021472 LDA TEMPS
3585      05622 011421 CPA PRADD END OF CURRENT STMT?
3586      05623 170402 RET YES
3587      05624 070517 LDA A,I READ CURRENT WORD
3588      05625 070452 SAM NXTP3 SKIP IF SIGN BIT SET
3589      05626 050065 AND 0PMSK
3590      05627 010116 CPA Q1OP QUOTE OPERATOR?
3591      05630 067612 JMP NXTP2 YES, PROCESS STRING CONSTANT

```

PAGE 75

9830—PRE-EXECUTION PROCESSING

3592 05631 010065 CPA OPMSK MULTI-CHARACTER-NON-FORMULA
OPERATOR?

3593 05632 067620 NXT1L JMP NXTP1 YES, NO OPERAND IN THIS WORD

3594 05633 060771 JSM GTOPP GET OPERAND

3595 05634 071210 SZA NXTP1 SKIP IF NULL

3596 05635 064356 JMP RETN3 RETURN TO P+3 WITH OPERAND
IN (A)

3597* PROCESS WORD WITH SIGN BIT SET

3598 05636 05016 NXTP3 AND B777 GET TYPE AND CHARACTER COUNT
IF CONSTANT

3599 05637 070137 LDB A

3600 05640 050040 AND B37 ISOLATE THE TYPE

3601 05641 010021 CPA .4 RESEQUENCABLE INTEGER?

3602 05642 067654 JMP NXTP4 YES

3603 05643 071350 SZA NXT1L SKIP IF PARAMETER

3604 05644 000135 ADA M4

3605 05645 071253 SAP NXT1L SKIP IF PARAMETER OR FUNCTION

3606 05646 074202 SBR 5 COMPUTE NUMBER OF

3607 05647 004022 ADB .5 WORDS TO SKIP IF

3608 05650 074042 SBR 2 CONSTANT

3609 05651 010132 CPA M1 INTEGER?

3610 05652 024012 LDB .1 YES, SKIP 1 WORD

3611 05653 067616 JMP NXTP1-2

3612 05654 045472 NXTP4 ISZ TEMPS READ THE

3613 05655 121472 LDA TEMPS,1 INTEGER

3614 05656 064357 JMP RETN2 RETURN TO P+2 WITH INTEGER
IN (A)

3615*

3616* ROUTINE TO FIND ALL RESEQUENCABLE LINE NUMBER REFERENCES IN A
PROGRAM

3617* EXIT: P+1 END OF PROGRAM FOUND

3618 05657 067620 JMP NXTP1 NO

3619* (A)=LINE NUMBER

3620* (TEMPS)=ADDRESS OF LINE NUMBER

3621*

3622 05660 031421 FNDAD STA PRADD SAVE STARTING ADDRESS

3623 05661 070517 LDA A,1 TYPING AID KEY?

3624 05662 070150 SZA FNDA2+2 RETURN IF YES

3625 05663 021421 FNDA2 LDA PRADD

3626 05664 011423 CPA PBPTR END OF PROGRAM?

3627 05665 170402 RET YES

3628 05666 063601 JSM NXTOP NO, CONTINUE SEARCH

3629 05667 067663 JMP FNDA2 END OF STATEMENT

3630 05670 064357 JMP RETN2 LINE NUMBER FOUND

3631 05671 063620 FNDA1 JSM NXTP1 CONTINUE SEARCH

3632 05672 067663 JMP FNDA2

3633 05673 064357 JMP RETN2

3634 05674 067671 JMP FNDA1

3635*

3636* ROUTINE TO CONVERT ALL STATEMENT REFERENCES INTO ADDRESS FORMAT

3637* THE ADDRESS OF A STATEMENT REPLACES THE LINE NUMBER AND

3638* THE SIGN BIT IS SET

3639*

3640 05675 021424 COMPL LDA PBUFF INITIALIZE TO START OF PROGRAM

3641 05676 063660 JSM FNDAD FIND FIRST LINE #

3642 05677 170402 RET

3643 05700 070312 SAM COMP1

3644 05701 061075 JSM FNDPS

3645 05702 067703 JMP *+1

3646 05703 164343 JMP E45-1,1 NOT IN PROGRAM

3647 05704 076053 SBP *+1,S FOUND SET SIGN BIT

PAGE 76

9830—PRE-EXECUTION PROCESS

```

3648 05705 135472      STB  TEMPS,I  REPLACE LINE NUMBER
3649 05706 063671 COMP1 JSM  FNDA1   CONTINUE SEARCH
3650 05707 170402      RET
3651 05710 067700      JMP  COMPL+3
3652*
3653*  ROUTINE TO CONVERT ALL STATEMENT REFERENCES INTO LINE NUMBER
                                           FORMAT
3654*
3655 05711 025424 DCOMP LDB  PBUFF
3656 05712 035415      STB  NXTDT  RESET DATA POINTERS
3657 05713 035414      STB  DSTRT
3658 05714 070742      SAR  16
3659 05715 031420      STA  PARMN  CLEAR PARAMETER NAME
3660 05716 076053      SBP  *+1,S  SET DECOMPILE FLAG
3661 05717 063514 DCOM1 JSM  SSTT+1  SAVE POINTERS
3662 05720 021421      LDA  PRADD
3663 05721 070113      SAP  *+2
3664 05722 067750      JMP  DCOMF  ALREADY DECOMPILED?
3665 05723 020166      LDA  FWAM  YES, DECOMPILE FUNCTIONS ONLY
3666 05724 011405 DCMF5 CPA  FWUP
3667 05725 067764      JMP  DCMF4  END OF KEY AREA?
3668 05726 070070      SIA  *+1  YES, NOW DECOMPILE MAINLINE
3669 05727 070137      LDB  A
3670 05730 070437      ADB  A,I
3671 05731 070070      SIA  *+1
3672 05732 035423 DCMF6 STB  PBPTR  COMPUTE START AND END OF KEY
3673 05733 063660      JSM  FNDAD  FIND ADDRESS WORDS
3674 05734 067744      JMP  DCMF2  END OF THIS PROGRAM
3675 05735 070233 DCMF1 SAP  DCMF3,C  SKIP IF ALREADY DECOMPILED
3676 05736 070517      LDA  A,I  READ LINE NUMBER
3677 05737 070073      SAP  *+1,C  CLEAR SECURE BIT
3678 05740 131472      STA  TEMPS,I  STORE LINE #
3679 05741 063671 DCMF3 JSM  FNDA1  FIND NEXT ADDRESS
3680 05742 067744      JMP  DCMF2
3681 05743 067735      JMP  DCMF1
3682*  END OF CURRENT PROGRAM
3683 05744 021423 DCMF2 LDA  PBPTR
3684 05745 011426      CPA  PBPTO  END OF MAINLINE?
3685 05746 067750      JMP  DCOMF  YES
3686 05747 067724      JMP  DCMF5  NO, CONTINUE KEYS
3687*  DECOMPILE FUNCTIONS IN SYMTAB BY SETTING THEIR ADDRESSES TO 0
                                           RESTORE PROGRAM POINTERS
3688 05750 063543 DCOMF JSM  RSTT
3689 05751 024040 DCMF1 LDB  B37
3690 05752 035470      STB  MASK  SET UP MASK FOR FUNCTION TYPE
3691 05753 035471      STB  STMP1
3692 05754 025406      LDB  VALUE
3693 05755 063374 DCMF2 JSM  SVALU  SEARCH SYMTAB FOR FUNCTION
3694 05756 170402      RET  END OF TABLE
3695 05757 004132      ADB  M1
3696 05760 070742      SAR  16
3697 05761 074557      STA  B,I  SET ADDRESS TO ZERO
3698 05762 004132      ADB  M1
3699 05763 067755      JMP  DCMF2  FIND NEXT FUNCTION
3700*
3701*  END OF KEY AREA
3702 05764 021425 DCMF4 LDA  PBUFO
3703 05765 025422      LDB  UKPTR

```

(197)

PAGE 77

9830—PRE-EXECUTION PROCESSING

3704	05766	076110	RZB	*+2	KEY MODE?
3705	05767	021424	LDA	PBUFF	NO, START OF MAINLINE IS IN PBUFF
3706	05770	025426	LDB	PBPTO	GET END OF MAINLINE
3707	05771	067732	JMP	DCMP6	
3708*					
3709*					
3710	00020		MDMA	EQU .3	TYPE CODE FOR DIM
3711	00017		MCOMA	EQU .2	
3712	00012		MDEFA	EQU .1	
3713	05772	164006	MREM	ABS 5-13777B	OP CODE 5 ON STANDARD PAGE
3714*					
3715	05773	000000	BSS	1	CHECKSUM WORD*****



PAGE 78

LIST ROUTINE

```

3717*
3718* THIS SUBROUTINE LISTS ALL OR
3719* A PORTION OF THE PROGRAM.
3720*
3721* ENTRY CONDITIONS:
3722* (A)=FWA OF LIST
3723* (B)=LWA+1 OF LIST
3724* PBUF=FWA OF PROGRAM
3725* PBPTR=LWA+1 OF PROGRAM
3726*
3727 060000 ORG 6000B
3728 060000 031453 LIST STA ARW SET FLAT FOR LIST
3729 060001 035460 STB TMP7 SAVE ENDING ADDRESS
3730 060002 074057 CPA B NULL LIST?
3731 060003 066005 JMP *+2 YES
3732 060004 061344 JSM CRLF1
3733 060005 021453 LDA ARW
3734 060006 031472 LISTI STA TEMPS SAVE STARTING ADDRESS
3735 060007 031476 STA TMP11 SET FLAG FOR BLANK AFTER MCOU
3736 060010 070742 SAR 16
3737 060011 031432 STA DRQST CLEAR INPUT MODE
3738* LIST NEXT LINE
3739 060012 160170 L1 JSM CLERA,I CLEAR BUFFER
3740 060013 025472 LDB TEMPS
3741 060014 015460 CPB TMP7 FINISHED
3742 060015 170402 RET YES
3743 060016 062036 JSM LISTS NO, PUT NEXT LINE IN BUFFER
3744 060017 025453 LDB ARW
3745 060020 076350 RZB L100 UP OR DOWN ARROW?
3746 060021 020040 LDA .31 YES
3747 060022 045623 ISZ PEND GUARANTEE ROOM FOR LAZY-T
3748 060023 061016 JSM ONEXT OUTPUT LAZY-T
3749 060024 020132 LDA M1
3750 060025 031430 STA LNINC FLAG REED FOR LIST
3751 060026 165462 JMP REEDL,I
3752 060027 060373 L100 JSM ICNT COMPUTE COUNT
3753 060030 061337 JSM NDWRT OUTPUT TO PRINTER
3754 060031 061344 JSM CRLF1
3755 060032 021442 LDA STPFL
3756 060033 010040 CPA .31 STOP KEY HIT?
3757 060034 167517 JMP CRCIA,I YES, ABORT
3758 060035 066012 JMP L1
3759*
3760* SUBROUTINE TO PUT NEXT LINE IN BUFFER
3761*
3762 060036 061007 LISTS JSM NEXTA FIND END OF LINE
3763 060037 035475 STB TMP10 SAVE END OF STMT
3764 060040 020236 LDA SLEND
3765 060041 031623 STA PEND SET END OF BUFFER POINTER
3766 060042 125472 LDB TEMPS,I
3767 060043 074073 SBP *-1,C CLEAR SECURE BIT
3768 060044 035431 STB CLINE
3769 060045 076650 RZB LLL TYPING AID KEY?
3770 060046 020100 LDA B52 YES, OUTPUT AN *
3771 060047 061016 JSM ONEXT
3772 060050 025460 LDB TMP7

```

PAGE 79

LIST ROUTINE

3773	06051	035475	STB	TMP10	SET UP END OF LINE POINTER
3774	06052	055472	DSZ	TEMPS	
3775	06053	121472	LDA	TEMPS,I	
3776	06054	000133	ADA	M2	
3777	06055	070744	SAL	1	GET CHARACTER COUNT
3778	06056	025472	LDB	TEMPS	
3779	06057	004017	ADB	.2	
3780	06060	074744	SBL	1	GET STARTING ADDRESS
3781	06061	066276	JMP	L28-3	OUTPUT STRING
3782*	OUTPUT STMT NUMBER AND TYPE				
3783	06062	160210	LLL	JSM	OUTIA,I
3784	06063	061015	JSM	OBLNK	
3785	06064	121472	LDA	TEMPS,I	
3786	06065	070213	SAP	*+4	SECURE?
3787	06066	020100	LDA	B52	YES, OUTPUT A ""
3788	06067	061016	JSM	ONEXT	
3789	06070	066277	JMP	L28-2	
3790	06071	060602	JSM	NEXT1-4	NO, GET PAGE AND TYPE
3791	06072	035474	STB	TMP9	STORE LAST ADDRESS OF PAGE
3792	06073	055472	DSZ	TEMPS	
3793	06074	121472	LDA	TEMPS,I	
3794	06075	045472	ISZ	TEMPS	
3795	06076	050107	AND	B1000	GET ASCII FLAG
3796	06077	031461	STA	TMP13	
3797	06100	021466	LDA	LOCLI	RECALL OP CODE
3798	06101	010132	CPA	M1	IMPLIED LET?
3799	06102	066117	JMP	L23	YES
3800	06103	070076	TCA		GET OP-CODE
3801	06104	004132	ADB	M1	GET STMT NAME TABLE ADDRESS
3802	06105	074437	ADB	B,I	
3803	06106	074744	SBL	1	COMPUTE TABLE ADDRESS
3804*	OUTPUT MULTI-CHARACTER OPERATOR				
3805	06107	062306	L6	JSM	MCOUT
3806	06110	021461	LDA	TMP13	OUTPUT NAME
3807	06111	070110	SZA	*+2	ASCII FORMAT?
3808	06112	066266	JMP	L25	YES
3809	06113	021476	LDA	TMP11	
3810	06114	070110	SZA	*+2	
3811	06115	061015	JSM	OBLNK	
3812	06116	035476	STB	TMP11	RESET BLANK FLAG
3813*	OUTPUT NEXT OPERAND				
3814	06117	121472	L23	LDA	TEMPS,I
3815	06120	050067	AND	TYPFL	ISOLATE OPERAND TYPE
3816	06121	070472	SAM	L4,C	SKIP IF CONSTANT OR FUNCTION
3817	06122	031477	STA	TMP12	SAVE TYPE
3818	06123	121472	LDA	TEMPS,I	
3819	06124	050065	AND	OPMSK	WAS THE PREVIOUS OPERATOR
3820	06125	010065	CPA	OPMSK	EXPANDED?
3821	06126	066152	JMP	L11	YES SKIP THIS FIELD
3822	06127	062302	JSM	GTOPD	
3823	06130	071710	SZA	*-2	SKIP IF NULL OPERAND
3824	06131	066221	JMP	L15	
3825*	OUTPUT CONSTANT				
3826	06132	010040	L4	CPA	.31
3827	06133	066255	JMP	L9	PREDEFINED FUNCTION?
3828	06134	000132	ADA	M1	YES

PAGE 80

LIST ROUTINE

3829	06135	031705	STA	FLOAT	SAVE FIX/FLOAT FLAG (0=FIX)
3830	06136	000133	ADA	M2	
3831	06137	070313	SAP	L12	SKIP IF INTEGER
3832	06140	160232	JSM	SNFLA,I	CONVERT
3833	06141	020133	LDA	M2	
3834	06142	031703	STA	PLACE	DECIMAL PLACES NOT SPECIFIED
3835	06143	062353	JSM	NMOUT	OUTPUT NUMBER FROM AR2
3836	06144	066152	JMP	L11	
3837*	OUTPUT INTEGER				
3838	06145	04572	ISZ	TEMPS	
3839	06146	125472	LDB	TEMPS,I	GET INTEGER
3840	06147	074133	SBP	*+2,C	COMPILED FORMAT?
3841	06150	074537	LDB	B,I	YES, READ LINE #
3842	06151	160210	JSM	OUTIA,I	OUTPUT INTEGER
3843*	*CHECK FOR END OF STATEMENT				
3844	06152	045472	ISZ	TEMPS	
3845	06153	021472	LDA	TEMPS	
3846	06154	011475	CPA	TMP10	
3847	06155	170402	RET		
3848*	OUTPUT NEXT OPERATOR				
3849	06156	121472	LDA	TEMPS,I	
3850	06157	050065	AND	OPMSK	
3851	06160	070442	SAR	10	
3852	06161	070510	SZA	L13	SKIP IF NULL OPERATOR
3853	06162	010012	CPA	1	QUOTE?
3854	06163	066213	JMP	L29	YES
3855	06164	024034	LDB	21	
3856	06165	060753	JSM	LIMIT	SINGLE CHARACTER OPERATOR?
3857	06166	066174	JMP	L13	NO
3858	06167	000123	ADA	FOPBS	
3859	06170	070517	LDA	A,I	
3860	06171	070342	SAR	8	GET ASCII FROM TABLE
3861	06172	061016	JSM	ONEXT	
3862	06173	066117	JMP	L23	GO OUTPUT NEXT OPERAND
3863*	OUTPUT MULTI-CHAR OPERATOR				
3864	06174	031703	STA	PLACE	SAVE OPERAND IN TEMP
3865	06175	010040	CPA	31	EXPANDED OP?
3866	06176	062302	JSM	GTOPD	YES, GET OP CODE
3867	06177	031704	STA	WIDTH	SAVE
3868	06200	070142	SAR	4	
3869	06201	031476	STA	TMP11	SET FLAG FOR BLANK
3870	06202	070110	SZA	*+2	OUTPUT BLANK IF
3871	06203	061015	JSM	OBLNK	OP > 17B
3872	06204	021703	LDA	PLACE	
3873	06205	024130	LDB	FROMA	GET STANDARD PAGE
3874	06206	010040	CPA	31	EXPANDED OP?
3875	06207	025474	LDB	TMP9	YES, GET PAGE FOR STMT
3876	06210	021704	LDA	WIDTH	
3877	06211	004135	ADB	M4	
3878	06212	066105	JMP	L6-2	COMPUTE TABLE ADDR AND OUTPUT
3879*	OUTPUT QUOTE FIELD				
3880	06213	020043	LDA	34	
3881	06214	061016	JSM	ONEXT	
3882	06215	062647	JSM	QUOTE	
3883	06216	020043	LDA	34	
3884	06217	061016	JSM	ONEXT	

PAGE 81

LIST ROUTINE

```

3885 06220 066153 JMP L11+1
3886* OUTPUT OPERAND NAME AND TYPE
3887 06221 021477 L15 LDA TMP12
3888 06222 010040 CPA .31 FUNCTION?
3889 06223 062251 JSM L16 YES
3890 06224 062302 L8 JSM GTOPD
3891 06225 000101 ADA B100
3892 06226 061016 JSM ONEXT OUTPUT LETTER NAME
3893 06227 025477 LDB TMP12
3894 06230 014040 CPB .31 FUNCTION?
3895 06231 066152 JMP L11 YES
3896 06232 020077 LDA DOLAR
3897 06233 014076 CPB B36 STRING
3898 06234 066241 JMP L19 YES
3899 06235 074146 RBR 4
3900 06236 074251 SLB L20 LETTER AND DIGIT?
3901 06237 021477 LDA TMP12 YES, OUTPUT DIGIT
3902 06240 040052 IOR .48
3903 06241 061016 L19 JSM ONEXT
3904 06242 066152 JMP L11
3905 06243 074642 L20 SBR 14
3906 06244 075710 SZB *-2 LETTER ONLY?
3907 06245 020055 LDA I NO
3908 06246 075551 SLB L19 INTEGER?
3909 06247 020073 LDA S NO — SPLIT
3910 06250 066241 JMP L19
3911* OUTPUT "FN"
3912 06251 020072 L16 LDA F
3913 06252 061016 JSM ONEXT OUTPUT F
3914 06253 023522 LDA N
3915 06254 065016 JMP ONEXT OUTPUT N AND RETURN TO L8
3916* OUTPUT PRE-DEFINED FUNCTION NAME
3917 06255 062302 L9 JSM GTOPD GET OP-CODE
3918 06256 031467 STA LOCL2
3919 06257 060605 JSM NEXT1-1 GET PAGE
3920 06260 004134 ADB M3
3921 06261 074437 ADB B,1
3922 06262 021467 LDA LOCL2
3923 06263 074744 SBL 1 COMPUTE TABLE ADDRESS
3924 06264 062306 JSM MCOUT
3925 06265 06617 JMP L23
3926* OUTPUT REM STATEMENT
3927 06266 021472 L25 LDA TEMPS
3928 06267 070076 TCA
3929 06270 001475 ADA TMP10
3930 06271 070744 SAL 1
3931 06272 000132 ADA M1 COMPUTE CHARACTER COUNT
3932 06273 025472 LDB TEMPS
3933 06274 074744 SBL 1
3934 06275 074070 SIB *+1 COMPUTE STARTING ADDRESS
3935 06276 062651 JSM STOUT OUTPUT STRING TO BUFFER
3936 06277 025475 LDB TMP10
3937 06300 035472 STB TEMPS
3938 06301 170402 L28 RET LINE COMPLETE
3939*
3940* GET OPERAND NAME

```

PAGE 82 LIST ROUTINE

```

3941*
3942 06302 121472 GTOPD LDA TEMPS,I
3943 06303 050115 AND B1777
3944 06304 070202 SAR 5
3945 06305 170402 RET

```

PAGE 83 OUTPUT TO BUFFER ROUTINES

```

3947*
3948* THIS SUBROUTINE SEARCHES THE
3949* MULTIPLE CHARACTER OP CODE
3950* TABLE. IT RETURNS WITH THE
3951* ASCII STRING IN THE SYNTAX
3952* BUFFER.
3953*
3954* ENTRY CONDITIONS
3955*
3956* (A)=OP CODE
3957* (B)=POINTER TO FIRST
3958* CHARACTER OF TABLE
3959*
3960*
3961 06306 031741 MCOUT STA TMP1 SAVE CODE
3962 06307 035740 STB TMP
3963 06310 070742 MC1 SAR 16
3964 06311 031743 STA TMP3 COUNT=0
3965 06312 061053 MC4 JSM GTMP
3966 06313 070306 RAR 7
3967 06314 070331 SLA MC2,C OP CODE?
3968 06315 070402 SAR 9 YES
3669 06316 050040 AND .31
3970 06317 011741 CPA TMP1 MATCH?
3971 06320 066324 JMP MC3 YES
3972 06321 066310 JMP MC1
3973 06322 045743 MC2 ISZ TMP3
3974 06323 066312 JMP MC4
3975 06324 021743 MC3 LDA TMP3 GET COUNT
3976 06325 025743 LDB TMP3
3977 06326 074056 CMB
3978 06327 005740 ADB TMP COMPUTE STARTING ADDRESS
3979 06330 066651 JMP STOUT

```

PAGE 84

OUTPUT TO BUFFER ROUTINES

3981*
 3982* THIS SUBROUTINE ROUNDS THE NUMBER
 3983* IN AR2. IF OVERFLOW OCCURS BUMP THE
 3984* EXPONENT AND SHIFT A ONE IN AS THE
 3985* MSD. B HAS THE LEADING ZERO COUNT
 3986* FOR THE ROUNDING CONSTANT.
 3987*
 3988*
 3989 06331 021744 RND2 LDA AR1
 3990 06332 031754 STA AR2
 3991 06333 170402 RET
 3992 06334 061256 RND JSM CLAR2
 3993 06335 020022 LDA .5
 3994 06336 174470 MRY ROUND OFF LSD-1
 3995 06337 170560 FXA
 3996 06340 170002 MOV SET A=E
 3997 06341 071410 SZA RND2
 3998 06342 024012 LDB .1
 3999 06343 174470 MRY SHIFT IN A ONE
 4000 06344 020013 LDA B400
 4001 06345 001744 ADA AR1
 4002 06346 031754 STA AR2 BUMP EXPONENT
 4003 06347 070340 AAR 8
 4004 06350 010060 CPA .100 EXPONENT OVERFLOW?
 4005 06351 064444 JMP XFAR2
 4006 06352 055777 DSZ RSPTR

NUMOUT ROUTINE

4007*
 4008* THIS ROUTINE OUTPUTS A NORMALIZED
 4009* NUMBER FROM AR2 INTO A BUFFER/
 4010*
 4011* PLACE=DIGITS FOLLOWING DECIMAL POINT
 4012* PLACE=-1 STANDARD FORMAT
 4013* PLACE=-2 LIST NUMBER
 4014* WIDTH=FIELD WIDTH
 4015* BADDR=OUTPUT BUFFER POINTER.
 4016* FLOAT=1 SPECIFIES FLOATING POINT
 4017*
 4018*
 4019*
 4020*
 4021*
 4022*
 4023 06353 060405 NMOUT JSM XFAR1
 4024 06354 024012 LDB .1
 4025 06355 035751 STB DIGIT INITIALIZE
 4026 06356 171400 NM3 MLS SHIFT NEXT DIGIT
 4027 06357 070450 SZA NM1 DIGIT IS ZERO?
 4028 06360 021703 LDA PLACE
 4029 06361 070070 SIA *+1
 4030 06362 072250 RZA NM2 STANDARD FORMAT?
 4031 06363 014027 CP8 .11
 4032 06364 066374 JMP NM4
 4033 06365 014030 CPB .12
 4034 06366 066374 JMP NM4
 4035 06367 035751 NM2 STB DIGIT
 4036 06370 074070 NM1 SIB *+1

504

PAGE 85 OUTPUT TO BUFFER ROUTINES

4037	06371	014031		CPB	.13	LAST DIGIT?
4038	06372	066374		JMP	*+2	YES
4039	06373	066356		JMP	NM3	NO
4040	06374	060444	NM4	JSM	XFAR2	
4041	06375	021705		LDA	FLOAT	
4042	06376	072110		RZA	*+2	FIXED OR FLOAT?
4043	06377	066467		JMP	OFXD	FIXED
4044*						
4045*						
4046*						
4047*						
4048*						
4049*						
4050*						
4051	06400	025703	OFLT	LDB	PLACE	
4052	06401	021416		LDA	MODE	
4053	06402	070151		SLA	*+3	SKIP IF PROG OR LIST
4054	06403	035741		STB	R	
4055	06404	066433		JMP	FLT2+1	
4056	06405	074513		SBP	FLT1	STANDARD FORMAT?
4057	06406	014132		CPB	M1	LIST NUMBER?
4058	06407	066414		JMP	FLT7	NO
4059	06410	021751		LDA	DIGIT	YES
4060	06411	000132		ADA	M1	
4061	06412	031741		STA	R	R=DIGIT-1
4062	06413	066435		JMP	FLT5	
4063	06414	024022	FLT7	LDB	.5	STD PRINTOUT---FLOAT 5
4064	06415	035741	FLT11	STB	R	
4065	06416	066432		JMP	FLT2	
4066	06417	035741	FLT1	STB	R	
4067	06420	076110		RZB	*+2	ANY RIGHT DIGITS?
4068	06421	004132		ADB	M1	NO, ALLOW ONE LESS CHARACTER
4069	06422	004024		ADB	.7	
4070	06423	074076		TCB		
4071	06424	005704		ADB	WIDTH	
4072	06425	074213		SBP	FLT4	
4073	06426	020077	FLT3	LDA	DOLAR	NO
4074	06427	025704		LDB	WIDTH	
4075	06430	067475		JMP	MOUT	
4076	06431	063474	FLT4	JSM	MOUT-1	
4077	06432	025741	FLT2	LDB	R	
4078	06433	004017		ADB	.2	
4079	06434	062334		JSM	RND	ROUND THE NUMBER
4080	06435	062637	FLT5	JSM	OSIGN	
4081	06436	061314		JSM	ODGIT	OUTPUT A DIGIT FROM AR2
4082	06437	021741		LDA	R	
4083	06440	070310		SZA	FLT8	ANY RIGHT DIGITS?
4084	06441	020051		LDA	POINT	
4085	06442	061016		JSM	ONEXT	YES, OUTPUT DECIMAL POINT
4086	06443	061314	FLT6	JSM	ODGIT	
4087	06444	055741		DSZ	R	DINC R
4088	06445	066443		JMP	FLT6	
4089	06446	020071	FLT8	LDA	E	
4090	06447	061016		JSM	ONEXT	
4091	06450	025754		LDB	AR2	
4092	06451	074340		ABR	8	GET EXPONENT

PAGE 86

OUTPUT TO BUFFER ROUTINES

4093	06452	020047		LDA	PLUS	
4094	06453	074153		SPB	FLT10	NEGATIVE EXPONENT?
4095	06454	020050		LDA	MINUS	YES
4096	06455	074076		TCB		
4097	06456	035743	FLT10	STB	INDEX	SAVE EXPONENT
4098	06457	061016		JSM	ONEXT	
4099	06460	025743		LDB	INDEX	
4100	06461	004141		ADB	M10	
4101	06426	074153		SBP	FLT9	EXPONENT LT 10?
4102	06463	020052		LDA	B60	
4103	06464	061016		JSM	ONEXT	
4104	06465	025743	FLT9	LDB	INDEX	
4105	06466	164210		JMP	OUTIA.I	OUTPUT EXPONENT

206

194

1,444,141

194

PAGE 87

OUTPUT TO BUFFER ROUTINES

4107*
 4108* OUTPUT A FIXED POINT NUMBER
 4109* IF IT WILL FIT IN THE SPECIFIED
 4110* FIELD. THE NORMALIZED NUMBER IS
 4111* IN REGISTER AR1.
 4112*
 4113*
 4114 06467 021754 OFXD LDA AR2
 4115 06470 070340 AAR 8
 4116 06471 070070 SIA *+1
 4117 06472 031742 STA L
 4118 06473 074742 SBR 16
 4119 06474 035750 STB RBASE
 4120 06475 070076 TCA
 4121 06476 070152 SAM RL4
 4122 06477 035742 STB L
 4123 06500 031750 STA RBASE
 4124 06501 001751 RL4 ADA DIGIT
 4125 06502 070113 SAP *+2
 4126 06503 070742 SAR 16 DIGIT GE EXP?
 4127 06504 031741 STA R NO, R=0
 4128 06505 025703 RL3 LDB PLACE R=DIGIT+RBASE-L
 4129 06506 014133 CPB M2
 4130 06507 066567 JMP NN2 LIST NUMBER?
 4131 06510 074313 SBP N2 YES
 4132 06511 025741 LDB R STANDARD FORMAT?
 4133 06512 014027 CPB .11 YES
 4134 06513 004133 ADB M2 11 RIGHT DIGITS?
 4135 06514 014061 CPB .10 YES, USE ONLY 9
 4136 06515 004132 ADB M1 10 RIGHT DIGITS?
 4137 06516 035741 N2 STB R YES, USE ONLY 9
 4138 06517 074076 TCB
 4139 06520 005750 ADB RBASE
 4140 06521 074076 TCB
 4141 06522 005742 ADB L
 4142 06523 004143 ADB M12
 4143 06524 074353 SBP N1
 4144 06525 004031 ADB .13
 4145 06526 062334 N4 JSM RND ROUND THE NUMBER
 4146 06527 066533 JMP N1
 4147 06530 025754 N6 LDB AR2
 4148 06531 074112 SBM N1
 4149 06532 045742 ISZ L L GE 0?
 4150 06533 021416 N1 LDA MODE YES, BUMP L
 4151 06534 025754 LDB AR2
 4152 06535 074340 ABR 8
 4153 06536 070311 SLA NN3
 4154 06537 004143 ADB M12 CALC MODE?
 4155 06540 074513 SBP NN YES
 4156 06541 005703 ADB PLACE OVERFLOW?
 4157 06542 004030 ADB .12 NO
 4158 06543 074313 SBP N7 UNDERFLOW?
 4159 06544 021703 NN3 LDA PLACE
 4160 06545 070313 SAP NN1 STANDARD FORMAT?
 4161 06546 004141 ADB M10
 4162 06547 074153 SBP NN EXPONENT GT 9?

PAGE 88

OUTPUT TO BUFFER ROUTINES

4163	06550	004030		ADB	.12	
4164	06551	074713	N7	SBP	NN2	OVERFLOW?
4165	06552	066400	NN	JMP	OFLT	YES
4166	06553	025741	NN1	LDB	R	
4167	06554	074130		SIB	*+2	R=0?
4168	06555	004012		ADB	.1	NO, B=R+2
4169	06556	021742		LDA	L	YES, B=R+1
4170	06557	070130		SIA	*+2	L=0?
4171	06560	000132		ADA	M1	NO
4172	06561	070037		ADB	A	YES, USE L=1
4173	06562	074076		TCB		
4174	06563	005704		ADB	WIDTH	
4175	06564	075312		SBM	NN	NUMBER FITS IN FIELD?
4176	06565	035461		STB	TMP13	
4177	06566	063474		JSM	MOUT-1	
4178	06567	062637	NN2	JSM	OSIGN	
4179	06570	021742		LDA	L	
4180	06571	025742		LDB	L	
4181	06572	004144		ADB	M13	
4182	06573	074112		SBM	NN10	MORE THAN 12 DIGITS?
4183	06574	020030		LDA	.12	YES, USE ONLY 12
4184	06575	031461	NN10	STA	TMP13	
4185	06576	070510		SZA	NN4	
4186	06577	061314	NN5	JSM	ODGIT	
4187	06600	055461		DSZ	TMP13	LAST LEFT DIGIT?
4188	06601	066577		JMP	NN5	NO
4189	06602	025742		LDB	L	
4190	06603	004143		ADB	M12	
4191	06604	074312		SBM	NN4+2	
4192	06605	074250		SZB	NN4+2	
4193	06606	020052		LDA	B60	
4194	06607	063475		JSM	MOUT	FILL WITH ZEROES
4195	06610	020052	NN4	LDA	B60	
4196	06611	061016		JSM	ONEXT	
4197	06612	021741		LDA	R	
4198	06613	072110		RZA	*+2	
4199	06614	066636	NN11	JMP	NN6	
4200	06615	020051		LDA	POINT	
4201	06616	061016		JSM	ONEXT	
4202	06617	025750		LDB	RBASE	
4203	06620	074117		LDA	B	
4204	06621	070076		TCA		
4205	06622	001741		ADA	R	
4206	06623	070153		SAP	*+3	
4207	06624	070742		SAR	16	
4208	06625	025741		LDB	R	
4209	06626	031741		STA	R	R=R-RBASE
4210	06627	020052		LDA	B60	
4211	06630	063475		JSM	MOUT	FILL WITH ZEROES
4212	06631	021741	NN7	LDA	R	
4213	06632	070210		SZA	NN6	
4214	06633	061314		JSM	ODGIT	
4215	06634	055741		DSZ	R	LAST RIGHT DIGIT?
4216	06635	066631		JMP	NN7	NO
4217	06636	170402	NN6	RET		YES, DONE
4218*						

508

196

1,444,141

196

PAGE 89 OUTPUT TO BUFFER ROUTINES

```

4220*
4221* THIS SUBROUTINE OUTPUTS THE MANTISSA SIGN
4222*
4223* MANTISSA NEGATIVE ---- UNCONDITIONALLY OUTPUT -
4224*
4225* MANTISSA POSITIVE
4226* LIST NUMBER ----- OUTPUT NOTHING
4227* OTHERWISE ----- OUTPUT BLANK
4228*
4229*
4230 06637 021754 OSIGN LDA AR2
4231 06640 070151 SLA OSI MANTISSA POSITIVE?
4232 06641 020050 LDA MINUS NO
4233 06642 065016 JMP ONEXT
4234 06643 021703 LDA PLACE
4235 06644 010133 CPA M2 LIST NUMBER?
4236 06645 170402 RET YES
4237 06646 065015 JMP OBLNK

```

PAGE 90 OUTPUT TO BUFFER ROUTINES

```

4239*
4240* OUTPUT A QUOTE STRING
4241*
4242 06647 063261 QUOTE JSM OTCHI GET CHAR COUNT AND ADDRESS
4243 06650 000000 BSS 1 CHECKSUM WORD****(QTCHI
                                         RETURNS TO P+2)
4244*
4245*
4246* OUTPUT ANY STRING
4247*
4248* ENTRY CONDITIONS:
4249*
4250* A=NUMBER OF CHARACTERS
4251* B=POINTER FIRST CHAR
4252*
4253 06651 031466 STOUT STA LOCL1 SAVE COUNT
4254 06652 070742 SAR 16
4255 06653 031701 STA NDISP ENABLE DISPLAY
4256 06654 020012 LDA 1
4257 06655 065332 JMP NPWR+2 DISABLE PRINT

```

009

PAGE 91

PRINT, DISP, AND WRITE COMMAND EXECUTION

```

4259*
4260* SUBROUTINE TO PLACE NEXT PRINT FIELD IN BUFFER
4261*
4262* RETURN P+1: END OF LINE (B)=0 FOR CRLF NOT SUPPRESSED
4263* P+2: ITEM IN BUFFER (BADDR)=BUFFER POINTER
4264*
4265*
4266 06656 063227 PRINN JSM CKEOL END OF LINE?
4267 06657 074742 SBR 16 YES CRLF NOT SUPPRESSED
4268 06660 170402 RET YES CRLF SUPPRESSED
4269 06661 063004 JSM PFOR1 SET UP 72 CHAR LINE
4270 06662 063240 JSM QTCHK QUOTE FIELD NEXT?
4271 06663 066674 JMP PR5 NO
4272 06664 021471 QOV LDA L2T2 RECALL COUNT
4273 06665 003523 ADA M72
4274 06666 070113 SAP *+2 MORE THAN 72 CHARS?
4275 06667 070742 SAR 16 NO
4276 06670 070076 TCA
4277 06671 001471 ADA L2T2 TRUNCATE TO 72
4278 06672 062651 JSM STOUT OUTPUT STRING
4279 06673 066731 JMP PR7
4280 06674 121472 PR5 LDA TEMPS,I
4281 06675 050070 AND OPDMK
4282 06676 012732 CPA TABOP TAB?
4283 06677 067040 JMP ETAB YES
4284 06700 021453 LDA PRFLG
4285 06701 010132 CPA M1
4286 06702 066706 JMP *+4 KEYBOARD "LET"
4287 06703 062773 JSM PFORM
4288 06704 061220 JSM FETCH+1 EVALUATE EXPRESSION
4289 06705 066707 JMP *+2
4290 06706 062773 JSM PFORM EVALUATE "LET"
4291 06707 025622 LDB TERM
4292 06710 076150 RZB *+3 CRLF GIVEN IN FUNCTION?
4293 06711 160170 JSM CLERA,I YES
4294 06712 035622 STB TERM
4295 06713 066721 JMP PR6 NO
4296* BUFFER OVERFLOW
4297 06714 025742 POV LDB TMP2
4298 06715 074110 SZB *+2
4299 06716 066664 JMP QOV OVERFLOW DURING STRING
4300 06717 020215 LDA TMPOP NO, RESTORE AR2
4301 06720 060445 JSM XFAR2+1
4302 06721 024132 PR6 LDB M1
4303 06722 021416 LDA MODE
4304 06723 070150 SZA PR11 SKIP IF PROG MODE
4305 06724 025452 LDB CPLAC
4306 06725 021451 LDA CTYPE
4307 06726 063065 PR11 JSM GNUM1 SET UP CALC FORMAT FLAGS
4308 06727 074742 SBR 16 PUT NUMBER IN BUFFER
4309 06730 035742 STB TMP2 SET FLAG FOR NUMBER
4310 06731 063052 PR7 JSM OVCK CHECK FOP OVERFLOW
4311 06732 100077 TABOP OCT 100077 (OVCK RETURNS TO P+2)
4312 06733 075052 SBM POV SKIP IF OVERFLOW
4313 06734 021472 LDA TEMPS
4314 06735 011421 CPA PRADD END OF LINE?

```

PAGE 92

PRINT, DISP, AND WRITE COMMAND EXECUTION

4315	06736	064357		JMP	RETN2	YES RETURN TO P+2
4316	06737	121472		LDA	TEMPS,I	NO, GET NEXT OPERATOR
4317	06740	070073		SAP	*+1,C	
4318	06741	070442		SAR	10	
4319	06742	000133		ADA	M2	SET UP COMMA: SEMICOLON FLAG
4320	06743	063027		JSM	FILL	OUTPUT TRAILING BLANKS
4321	06744	064357		JMP	RETN2	
4322*						
4323*						
4324*	PRINT DISP AND WRITE(N,*) ROUTINE					
4325*						
4326	06745	020146	EPRIN	LDA	M15	SET UP SELECT CODE 15
4327	06746	070076	EDISP	TCA		NEG=DISP, -1=LET
4328	06747	031453		STA	PRFLG	SELECT CODE=PRINT OR WRITE
4329	06750	025416	EWRI1	LDB	MODE	
4330	06751	076150		RZB	*+3	SKIP IF KEYBOARD MODE
4331	06752	021622		LDA	TERM	
4332	06753	072150		RZA	*+3	SKIP IF BUFFER IS PARTIALLY FILLED
4333	06754	160170		JSM	CLERA,I	INITIALIZE BUFFER
4334	06755	035622		STB	TERM	INITIALIZE CRLF SUPPRESSED FLAG
4335	06756	063227		JSM	CKEOL	NULL LIST?
4336	06757	066765		JMP	PR2-1	
4337	06760	066765		JMP	PR2-1	YES, GIVE CRLF
4338	06761	062656	PR3	JSM	PRINN	OUTPUT NEXT FIELD TO BUFFER
4339	06762	066766		JMP	PR2	END OF LINE
4340	06763	063450		JSM	OLINE	PRINT THE LATEST FIELD
4341	06764	066761		JMP	PR3	
4342*						
4343	06765	074742		SBR	16	
4344	06766	035622	PR2	STB	TERM	SET CRLF SUPPRESSED FLAG
4345	06767	076150		RZB	PR17	SKIP IF SUPPRESSED
4346	06770	063007		JSM	PRCHK	DISP?
4347	06771	061344		JSM	CRLF1	OUTPUT A CRLF
4348	06772	164172	PR17	JMP	XEC4A,I	
4349*						
4350*	SAVE PRINT FLAGS AND CALL FORMX					
4351*						
4352	06773	025453	PFORM	LDB	PRFLG	
4353	06774	060563		JSM	SLWST	
4354	06775	025447		LDB	SLCOD	
4355	06776	060563		JSM	SLWST	
4356	06777	160201		JSM	FORMA,I	EVALUATE FORMULA
4357	07000	060567		JSM	UNSTK	
4358	07001	031447		STA	SLCOD	RESTORE SELECT CODE
4359	07002	060567		JSM	UNSTK	
4360	07003	031453		STA	PRFLG	RESTORE PRINT FLAG
4361	07004	023521	PFOR1	LDA	LN72	
4362	07005	031623		STA	PEND	SET UP 72 CHAR LINE
4363	07006	170402		RET		

211

PAGE 93

PRINT; DISP; AND WRITE COMMAND EXECUTION

```
4365 07007 025450 PRCHK LDB PMODE
4366 07010 074310 SZB PRCHR PRINT ALL?
4367 07011 025453 LDB PRFLG NO
4368 07012 074213 SBP PRCHR PRINT STMT?
4369 07013 064357 JMP RETN2 NO
4370* THIS SUBROUTINE FILLS A COMMA
4371* OR SEMICOLON FIELD WITH BLANKS/
4372* IF THE CURRENT LINE IS OVERFLOWED
4373* IT IS OUTPUT AND A CRLF IS DONE/
4374*
4375* ENTRY CONDITIONS:
4376*
4377* A=0 IF COMMA FIELD
4378* TMP2=0 IF QUOTE FIELD WAS PRINTED
4379*
4380 07014 021742 FILL1 LDA TMP2
4381 07015 070110 SZA *+2 SKIP IF NUMERIC
4382 07016 170402 PRCHR RET
4383 07017 061015 JSM OBLNK INSURE AT LEAST 1 BLANK
4384 07020 025625 LDB FIRST NO
4385 07021 074076 TCB
4386 07022 005400 ADB BADDR GET OUTPUT COUNT
4387 07023 004137 ADB M6
4388 07024 074512 SBM FILL2
4389 07025 004134 ADB M3
4390 07026 067024 JMP *-2
4391 07027 073250 FILL RZA FILL1 COMMA FIELD?
4392 07030 063014 JSM FILL1 YES, FIRST OUTPUT ":", SPACING
4393 07031 025400 LDB BADDR
4394 07032 007520 ADB MSLPT GET OUTPUT COUNT
4395 07033 075150 SZB PRCHR SKIP IF EXACT
4396 07034 004146 ADB M15
4397 07035 075713 SBP *-2
4398 07036 074076 FILL2 TCB
4399 07037 067474 JMP MOUNT-1 OUTPUT BLANKS
4400*
4401*
4402* TAB EXECUTION
4403*
4404*
4405 07040 045472 ETAB ISZ TEMPS
4406 07041 062773 JSM PFORM
4407 07042 061220 JSM FETCH+1
4408 07043 160226 JSM FLTFA,1 TRUNCATE TO INTEGER
4409 07044 064357 JMP RETN2 OVERFLOW
4410 07045 075752 SBM *-1 NEGATIVE NUMBER?
4411 07046 074076 TCB
4412 07047 005400 ADB BADDR COMPUTE # OF BLANKS NEEDED
4413 07050 007520 ADB MSLPT
4414 07051 063036 JSM FILL2
4415 07052 025400 OVCK LDB BADDR
4416 07053 015623 CPB PEND OVERFLOW?
4417 07054 067056 JMP ET1 YES
4418 07055 064357 JMP RETN2
4419 07056 160170 ET1 JSM CLERA,1
4420 07057 063007 JSM PRCHK PRINT?
```




PAGE 94

PRINT; DISP; AND WRITE COMMAND EXECUTION

4421	07060	061344		JSM	CRLF1	YES
4422	07061	024062		LDB	FLGBT	SET (B) NEGATIVE FOR OVERFLOW
4423	07062	035622		STB	TERM	SET CRLF SUPRESSED FLAG
4424	07063	064357		JMP	RETN2	
4425*						
4426	07064	160206	TABER	JSM	ERRA,1	ERROR IF CALLED FROM WRITE STMT
4427*						
4428*						
4429	07065	035703	GNUM1	STB	PLACE	
4430	07066	031705		STA	FLOAT	
4431	07067	024215	GNUM	LDB	TMPOP	
4432	07070	060406		JSM	XFAR1+1	SAVE RESULT
4433	07071	066353		JMP	NMOUT	PUT NUMBER IN BUFFER

PAGE 95

PRINT, DISP, AND WRITE COMMAND EXECUTION

4435*
 4436* WRITE STATEMENT EXECUTION
 4437*
 4438 07072 023447 EWRT LDA FMTOP
 4439 07073 063366 JSM SFMT SET UP DEVICE AND FORMAT CODES
 4440 07074 066750 JMP EWRI1 NO FORMAT SPECIFIED
 4441 07075 063227 JSM CKEOL NULL LIST?
 4442 07076 000000 BSS 1 ***CHECKSUM***IMPOSSIBLE STATE
 4443 07077 035472 STB TEMPS YES; SET TEMPS=PRADD
 4444 07100 063303 WRIT2 JSM TYPEC GET NEXT FORMAT SPEC
 4445 07101 067113 JMP WRITN END OF FORMAT
 4446 07102 067122 JMP WRITQ QUOTE FIELD
 4447 07103 074742 SBR 16 SET FLAG FOR FIXED
 4448 07104 060563 JSM SLWST STACK FLAG: 0=F, +=E, -=B
 4449 07105 067125 JMP WRITF (CODE 5 IS NOT USED)
 4450 07106 077713 SBP *-2,S SET B FLAG
 4451 07107 067117 JMP WRITX X;
 4452* END OF FORMAT STATEMENT
 4453 07110 063356 JSM FSPR /;UPDATA FORMAT POINTER
 4454 07111 061344 WRITS JSM CRLF1 /
 4455 07112 067100 JMP WRIT2
 4456 07113 075713 WRITN SBP WRITS ANY NUMERIC SPECS FOUND?
 4457 07114 074742 SBR 16
 4458 07115 035455 STB FORMT CLEAR FORMAT NESTING FLAG
 4459 07116 066750 JMP EWRI1 NO, GO TO STANDARD FORMAT
 4460* X FIELD
 4461 07117 063356 WRITX JSM FSPR UPDATE FORMAT POINTER
 4462 07120 020012 LDA .1
 4463 07121 027124 LDB BLNKA
 4464* QUOTE FIELD
 4465 07122 061337 WRITQ JSM NDWRT OUTPUT TO PRINTER
 4466 07123 067100 JMP WRIT2
 4467 07124 000103 BLNKA DEF .32+.32+1
 4468*
 4469* OUTPUT F,E, OR B FIELD
 4470 07125 025466 WRITF LDB LOCL1
 4471 07126 060563 JSM SLWST STACK FORMAT STMT POINTER
 4472 07127 063227 WRIT4 JSM CKEOL END OF WRITE STMT?
 4473 07130 074742 SBR 16 YES, CRLF NOT SUPRESSED
 4474 07131 067222 JMP WRIT8-2 YES
 4475 07132 063240 JSM QICLK QUOTE NEXT IN WRITE?
 4476 07133 067136 JMP WRIT5 NO
 4477 07134 061337 JSM NDWRT YES, PRINT IT
 4478 07135 067127 JMP WRIT4
 4479 07136 062773 WRIT5 JSM PFORM SAVE FLAGS AND
 4480 07137 061220 JSM FETCH+1 EVALUATE FORMULA
 4481 07140 060567 JSM UNSTK RECALL FORMAT STMT POINTER
 4482 07141 031466 STA LOCL1
 4483 07142 060567 JSM UNSTK RECALL E,F, OR B FLAG
 4484 07143 031705 STA FLOAT
 4485 07144 070252 SAM WRITB SKIP IF B
 4486 07145 045466 ISZ LOCL1
 4487 07146 121466 LDA LOCL1,1 GET FIELD WIDTH SPEC
 4488 07147 045466 ISZ LOCL1
 4489 07150 045466 ISZ LOCL1
 4490 07151 025456 WRITB LDB FORMS



PAGE 96

PRINT; DISP; AND WRITE COMMAND EXECUTION

4491	07152	074073	SBP	*+1,C	
4492	07153	035456	STB	FORMS	SET NUMERIC SPEC FOUND FLAG
4493	07154	125466	LDB	LOCL1,I	GET DECIMAL SPEC
4494	07155	063356	JSM	FSPR	UPDATE FORMAT POINTERS
4495	07156	031704	STA	WIDTH	
4496	07157	070353	SAP	WRIT7	SKIP IF E OR F
4497*	OUTPUT B FIELD CHARACTER				
4498	07160	160225	JSM	FLTRA,I	CONVERT TO ROUNDED INTEGER
4499	07161	074742	SBR	16	OVERFLOW: USE 0
4500	07162	074117	LDA	B	
4501	07163	050105	AND	B377	MASK TO 8 BITS
4502	07164	061353	JSM	TTY	OUTPUT SINGLE CHAR
4503	07165	067216	JMP	WRIT6	
4504*	OUTPUT E OR F NUMBER				
4505	07166	035703	WRIT7	STB	PLACE
4506	07167	021412	LDA	LSTPT	START TEMP BUFFER
4507	07170	070070	SIA	*+1	
4508	07171	070744	SAL	1	
4509	07172	031400	STA	BADDR	ABOVE LOW STACK
4510	07173	031706	STA	SBPTR	
4511	07174	021413	LDA	HSTPT	
4512	07175	070744	SAL	1	
4513	07176	031623	STA	PEND	SET END OF BUFFER AT H-STACK
4514	07177	021416	LDA	MODE	
4515	07200	031641	STA	ST+1	
4516	07201	070742	SAR	16	
4517	07202	031416	STA	MODE	SET FOR PROGRAM FORMAT
4518	07203	063067	JSM	GNUM	OUTPUT NUMBER TO BUFFER
4519	07204	021641	LDA	ST+1	
4520	07205	031416	STA	MODE	
4521	07206	021400	LDA	BADDR	
4522	07207	011623	CPA	PEND	OVERFLOW?
4523	07210	064561	JMP	MER8	
4524	07211	025706	LDB	SBPTR	
4525	07212	060374	JSM	WBUFF	COMPUTER CHAR COUNT
4526	07213	061337	JSM	NDWRT	PRINT THE BUFFER
4527	07214	025625	LDB	FIRST	
4528	07215	035400	STB	BADDR	RESET BADDR
4529	07216	063227	WRIT6	JSM	CKEOL
4530	07217	074742	SBR	16	END OF STMT?
4531	07220	067224	JMP	WRIT8	YES, CRLF NOT SUPRESSED
4532	07221	067100	JMP	WRIT2	YES
4533*	END OF WRITE STMT				
4534	07222	055412	DSZ	LSTPT	
4535	07223	055412	DSZ	LSTPT	UNSTACK FLAGS
4536	07224	070742	WRIT8	SAR	16
4537	07225	031455	STA	FORMT	SET FORMAT COMPLETE FLAG
4538	07226	066767	JMP	PR2+1	

015

PAGE 97

PRINT; DISP; AND WRITE COMMAND EXECUTION

```

4540*
4541* CHECK FOR END OF OUTPUT LIST
4542*   P+1: TEMPS=PRADD
4543*   P+2: NULL OPERAND AND TEMPS+1=PRADD
4544*   P+3: ALL OTHER
4545*
4546 07227 025472 CKEOL LDB TEMPS
4547 07230 015421 CPB PRADD END OF STMT?
4548 07231 170402 RET
4549 07232 060771 JSM GTOPP
4550 07233 072210 RZA CKEO1 NULL OPERAND?
4551 07234 074070 SIB *+1 YES
4552 07235 015421 CPB PRADD TEMPS+1=PRADD?
4553 07236 064357 JMP RETN2 YES
4554 07237 064356 CKEO1 JMP RETN3 NO
4555*
4556* CHECK TO SEE IF NEXT OPERATOR IS A QUOTE FIELD
4557*   P+1: QUOTE NOT NEXT
4558*   P+2: QUOTE NEXT; (A)=LENGTH; (B)=POINTER; TEMPS UPDATED
4559*
4560 07240 121472 QTCHK LDA TEMPS,I
4561 07241 050065 AND OPMSK CURRENT OPERATOR
4562 07242 010116 CPA QTOP A QUOTE?
4563 07243 067261 JMP QTCH1 YES
4564 07244 060771 JSM GTOPP
4565 07245 072410 RZA QTCH3 NULL OPERAND?
4566 07246 045472 ISZ TEMPS YES, GET NEXT OPERATOR
4567 07247 121472 LDA TEMPS,I
4568 07250 050065 AND OPMSK
4569 07251 010116 CPA QTOP QUOTE?
4570 07252 067261 JMP QTCH1 YES
4571 07253 055472 DSZ TEMPS NO, RESTORE TEMPS
4572 07254 170402 RET RETURN P+1
4573 07255 050040 QTCH3 AND B37
4574 07256 010076 CPA B36 STRING VARIABLE?
4575 07257 067276 JMP QTCH2 YES
4576 07260 170402 RET NO, RET P+1
4577 07261 121472 QTCH1 LDA TEMPS,I
4578 07262 050104 AND B177 GET CHAR COUNT
4579 07263 031471 STA L2T2 SAVE
4580 07264 045472 ISZ TEMPS
4581 07265 025472 LDB TEMPS
4582 07266 074744 SBL 1 GET STARTING CHARADDRESS
4583 07267 035742 STB TMP2 SAVE
4584 07270 070037 ADB A
4585 07271 074070 SIB *+1
4586 07272 074002 SBR 1
4587 07273 035472 STB TEMPS POINT TEMPS PAST END OF STRING
4588 07274 025742 LDB TMP2
4589 07275 064357 JMP RETN2
4590 07276 020121 QTCH2 LDA SBLOC
4591 07277 060613 JSM FPAGE FIND STRING BLOCK PAGE
4592 07300 074117 LDA B
4593 07301 050065 AND OPMSK GET STARTING ADDRESS OF PAGE
4594 07302 070737 JMP A,I JUMP TO STRING PAGE
4595 00121 SBLOC EQU B4000 OP CODE 2

```

PAGE 98

PRINT; DISP; AND WRITE COMMAND EXECUTION

```

4597*
4598*   SUBROUTINE TO GET NEXT FORMAT SPEC
4599*
4600  07303  021454  TYPEC   LDA  RCOUT
4601  07304  072450        RZA  FSP1
4602  07305  025455        LDB  FORMT
4603  07306  015457        CPB  FORME
4604  07307  067353        JMP  FSPN
4605*   SET UP NEW REPEAT COUNT
4606  07310  121455        LDA  FORMT,I
4607  07311  045455        ISZ  FORMT
4608  07312  070253        SAP  FSP2
4609  07313  121455        LDA  FORMT,I
4610  07314  045455        ISZ  FORMT
4611  07315  000132  FSP1  ADA  MI
4612  07316  031454        STA  RCOUT
4613*   GET FORMAT SPEC; (FORMT) POINTS TO TYPE WORD
4614  07317  121455  FSP2  LDA  FORMT,I
4615  07320  050065        AND  OPMSK  READ TYPE
4616  07321  025777        LDB  RSPTR
4617  07322  004132        ADB  MI
4618  07323  035401        STB  RSAVE  SAVE ADDRESS OF RETURN
4619  07324  025455        LDB  FORMT
4620  07325  035466        STB  LOCL1
4621  07326  010116        CPA  QTOP  QUOTE FIELD?
4622  07327  067340        JMP  FSPQ  YES
4623  07330  121455        LDA  FORMT,I
4624  07331  070202        SAR  5
4625  07332  050040        AND  B37
4626  07333  070137        LDB  A
4627  07334  105401        ADB  RSAVE,I
4628  07335  004132        ADB  MI
4629  07336  135401  BASE  STB  RSAVE,I
4630  07337  170402        RET
4631  07340  121466  FSPQ  LDA  LOCL1,I
4632  07341  050104        AND  B177  GET CHAR COUNT
4633  07342  031467        STA  LOCL2  SAVE
4634  07343  074070        SIB  *+1
4635  07344  074744        SBL  1  COMPUTE STARTING CHARACTER
                                         ADDRESS
4636  07345  074017        ADA  B
4637  07346  000132        ADA  MI
4638  07347  070002        SAR  1
4639  07350  031466        STA  LOCL1  COMPUTE AND SAVE ENDING WORD
                                         ADDRESS
4640  07351  145401        ISZ  RSAVE,I  SET TO RETURN TO P+2
4641  07352  067357        JMP  FSPR+1
4642*   END OF FORMAT STMT
4643  07353  025456  FSPN  LDB  FORMS  RESET TO START OF SMT
4644  07354  035455        STB  FORMT
4645  07355  170402        RET
4646*
4647*
4648*
4649  07356  031467  FSPR  STA  LOCL2  SAVE (A)
4650  07357  021454        LDA  RCOUT
4651  07360  072210        RZA  FSPR1  END OF REPEAT COUNT?
4652  07361  021466        LDA  LOCL1  YES, UPDATE

```

PAGE 99 PRINT, DISP, AND WRITE COMMAND EXECUTION

4653	07362	070070		SIA	*+1	
4654	07363	031455		STA	FORMAT	FORMAT POINTER
4655	07364	021467	FSPR1	LDA	LOCL2	RESTORE (A)
4656	07365	170402		RET		

PAGE 100 PRINT, DISP, AND WRITE COMMAND EXECUTION

4658*						
4659*						SET UP FORMAT STMT POINTERS FOR WRITE OR ENTER
4660*						
4661	07366	031640	SFMT	STA	ST	SAVE FORMAT OP CODE
4662	07367	045472		ISZ	TEMPS	
4563	07370	061217		JSM	FETCH	EVALUATE SELECT CODE
4664	07371	160225		JSM	FLTRA,I	CONVERT TO INTEGER
4665	07372	160206	SCERR	JSM	ERRA,I	OVERFLOW
4666	07373	074117		LDA	B	
4667	07374	075710		SZB	*-2	
4668	07375	074142		SBR	4	
4669	07376	077610		RZB	*-4	CHECK FOR 1<SC<15
4670	07377	031453		STA	PRFLG	SAVE SELECT CODE
4671	07400	070204		SAL	12	
4672	07401	031447		STA	SLCOD	
4673	07402	035454	PLOTE	STB	RCOUT	SET REPEAT COUNT TO 0
4674	07403	125472		LDB	TEMPS,I	
4675	07404	045472		ISZ	TEMPS	
4676	07405	074152		SBM	*+3	FORMAT SPECIFIED
4677	07406	045472		ISZ	TEMPS	
4678	07407	170402		RET		NO, RETURN P+1
4679	07410	121472		LDA	TEMPS,I	YES
4680	07411	070673		SAP	SFMT2,C	
4681	07412	025455	SFMT1	LOB	FORMAT	ERROR IF STATEMENTS USING
4682	07413	074110		SZB	*+2	FORMATS NESTED?
4683	07414	160206	FNEST	JSM	ERRA,I	YES
4684	07415	070137		LDB	A	
4685	07416	000017		ADA	.2	
4686	07417	031455		STA	FORMAT	SET CURRENT FMT SPEC POINTER
4687	07420	072053		SAP	*+1,S	SET FLAG — NO NUMERIC SPEC YET
4688	07421	031456		STA	FORMS	
4689	07422	061007		JSM	NEXTA	
4690	07423	035457		STB	FORME	SET POINTER TO END OF FMT STMT
4691	07424	045472		ISZ	TEMPS	
4692	07425	064357		JMP	RETN2	
4693*						FIND AND VERIFY FORMAT STMT
4694	07426	061075	SFMT2	JSM	FNDPS	FIND STATEMENT
4695	07427	067430		JMP	*+	
4696	07430	164343	NOFMT	JMP	E45-1,I	NOT FOUND
4697	07431	076053		SBP	*+1,S	SET FLAG
4698	07432	135472		STB	TEMPS,I	STORE ADDRESS IN PLACE OF STMT #
4699	07433	021476		LDA	TEMP3	
4700	07434	070070		SIA	*+1	
4701	07435	070537		LDA	A,I	GET OP CODE
4702	07436	070070		SIA	*+1	
4703	07437	070517		LDA	A,I	GET PAGE
4704	07440	050065		AND	OPMSK	
4705	07441	074442		SBR	10	
4706	07442	070037		ADB	A	COMBINE
4707	07443	021476		LDA	TEMP3	RECALL STMT ADDRESS
4708	07444	015640		CPB	ST	IS THIS A FORMAT STMT?
4709	07445	067412		JMP	SFMT1	YES
4710	07446	067430		JMP	NOFMT	NO, ERROR
4711*						
4712	07447	012025	FMTOP	OCT	12025	

PAGE 101

PRINT: DISP: AND WRITE COMMAND EXECUTION

4714* THIS ROUTINE TAKES A RECORD FROM
 4715* THE REFRESH BUFFER AND OUTPUTS IT
 4716* ON THE PRINTER.
 4717*
 4718*

4719	07450	021625	OLINE	LDA FIRST	
4720	07451	070076		TCA	
4721	07452	001400		ADA BADDR	GET CHAR COUNT
4722	07453	073410		SZA OLI	BUFFER EMPTY?
4723	07454	063007		JSM PRCHK	PRINT?
4724	07455	067457		JMP *+2	YES
4725	07456	067464		JMP OL2	NO
4726	07457	025625		LDB FIRST	YES
4727	07460	061337		JSM NDWRT	
4728	07461	021400	OL3	LDA BADDR	
4729	07462	031625		STA FIRST	
4730	07463	170402	OL1	RET	
4731	07464	024030	OL2	LDB .12	
4732	07465	035705		STB FLOAT	SAVE COUNT
4733	07466	163516	EW2	JSM REFRA.I	
4734	07467	025443		LDB KEYFG	HAS A KEY BEEN ENTERED?
4735	07470	075453		SBP OL3	
4736	07471	055705		DSZ FLOAT	
4737	07472	067466		JMP EW2	
4738	07473	067461		JMP OL3	

4739*
 4740* THIS ROUTINE OUTPUTS THE CODE.
 4741* IN A B TIMES
 4742*
 4743*

4744	07474	020041		LDA .32	
4745	07475	074412	MOUT	SBM MO1	COUNT NEGATIVE
4746	07476	074350		SZB MO1	OR ZERO?
4747	07477	035740		STB TMP	NO, SAVE COUNT
4748	07500	031752		STA TMP6A	
4749	07501	021752		LDA TMP6A	
4750	07502	061016	MO2	JSM ONEXT	
4751	07503	055740		DSZ TMP	
4752	07504	067501		JMP MO2-1	
4753	07505	170402	MO1	RET	YES

219

PAGE 102

PRINT: DISP: AND WRITE COMMAND EXECUTION

```
4755*  
4556* WAIT EXECUTION  
4757*  
4758 07506 061217 EWAIT JSM FETCH GET COUNT  
4759 07507 160226 JSM FLTFA,I CONVERT TO INTEGER  
4760 07510 164172 EWI JMP XEC4A,I OVERFLOW  
4761 07511 075752 SBM EW1  
4762 07512 074142 SBR 4 DIVIDE BY 16  
4763 07513 075650 SZB EW1  
4764 07514 063465 JSM EW2-1 REFRESH DISPLAY  
4765 07515 164172 JMP XEC4A,I
```

```
4767 07516 002562 REFRA DEF REFRS  
4768 07517 002512 CRCIA DEF CRCI  
4769 072520 174456 MSLPT ABS -RBUF-RBUF  
4770 07521 003433 LN72 DEF RBUF+RBUF+73  
4771 07522 000116 N OCT 116  
4772 07523 177670 M72 DEC -72
```


PAGE 103

SYNTAX TABLES

4774*

4775* STATEMENT SYNTAX JUMP TABLE

4776*

4777	07524	174556	ABS	RITE-*	
4778	07525	171422	ABS	ENDS-*	GRAD
4779	07526	171421	ABS	ENDS-*	DEG
4780	07527	171420	ABS	ENDS-*	RAD
4781	07530	174541	ABS	LETS-*	
4782	07531	173760	ABS	FMTS-*	
4783	07532	174554	ABS	PRINS-*	DISP
4784	07533	006725	ABS	READS-*	INPUT
4785	07534	006732	ABS	RSTRS-*	
4786	07535	174551	ABS	PRINS-*	
4787	07536	006722	ABS	READS-*	
4788	07537	173743	ABS	DATAS-*	
4789	07540	002741	ABS	WAITS-*	
4790	07541	174070	ABS	STOPS-*	
4791	07542	171405	ABS	ENDS-*	
4792	07543	002761	ABS	RETNS-*	
4793	07544	174011	ABS	GOTOS-*	GOSUB
4794	07545	000147	ABS	NEXTS-*	
4795	07546	002714	ABS	FORS-*	
4796	07547	000151	ABS	IFS-*	
4797	07550	174005	ABS	GOTOS-*	
4798	07551	006721	ABS	REMS-*	
4799	07552	002731	ABS	DEFS-*	
4800	07553	006660	ABS	COMS-*	
4801	07554	006702	ABS	DIMS-*	DIM
4802	07555	174446	ABS	ILETS-*	

4803*

4804* STATEMENT NAME TABLE

4805*

4806	07556	000241	SYN13	OCT	00241	IMPLIED LET OP CODE 100001
4807	07557	046105		ASC	1,LE	
4808	07560	052266		OCT	52266	LET; OP CODE 110110
4809	07561	042111		ASC	1,DI	
4810	07562	046602		OCT	046602	DIM;000010
4811	07563	041517		ASC	1,CO	
4812	07564	046603		OCT	046603	COM;000011
4813	07565	042105		ASC	1,DE	
4814	07566	043204		OCT	043204	DEF;000100
4815	07567	051105		ASC	1,RE	
4816	07570	046645		OCT	046645	REM;100101
4817	07571	043517		ASC	2,GOTO	
	07572	052117				
4818	07573	123111		OCT	123111	GOTO;100110
4819	07574	043247		OCT	043247	IF;100111
4820	07575	047105		ASC	2,NEXT	
	07576	054124				
4821	07577	104507		OCT	104507	NEXT;001001
4822	07600	047523		ASC	2,OSUB	
	07601	052502				
4823	07602	105122		OCT	105122	GOSUB;001010
4824	07603	042524		ASC	2,ETUR	
	07604	052522				
4825	07605	047213		OCT	047213	RETURN;001011

PAGE 104

SYNTAX TABLES

4826	07606	042516	ASC	1,EN	
4827	07607	042254	OCT	042254	END;101100
4828	07610	051524	ASC	2,STOP	
	07611	047520			
4829	07612	126527	OCT	126527	STOP;101101
4830	07613	040511	ASC	1,AI	
4831	07614	052216	OCT	052216	WAIT;001110
4832	07615	042101	ASC	2,DATA	
	07616	052101			
4833	07617	107522	OCT	107522	DATA;001111
4834	07620	042501	ASC	1,EA	
4835	07621	042220	OCT	042220	READ;010000
4836	07622	050122	ASC	2,PRIN	
	07623	044516			
4837	07624	052261	OCT	052261	PRINT;110001
4838	07625	042111	ASC	2,DISP	
	07626	051520			
4839	07627	132111	OCT	132111	DISP;110100
4840	07630	047120	ASC	2,NPUT	
	07631	052524			
4841	07632	111506	OCT	111506	INPUT;010011
4842	07633	047522	ASC	2,ORMA	
	07634	046501			
4843	07635	052225	OCT	52225	FORMAT;010101
4844	07636	043117	ASC	1,FO	
4845	07637	051210	OCT	51210	FOR;001000
4846	07640	051105	ASC	3,RESTOR	
	07641	051524			
	07642	047522			
4847	07643	042662	OCT	42662	RESTORE;110010
4848	07644	051101	ASC	1,RA	
4849	07645	042267	OCT	42267	RAD;110111
4850	07646	042105	ASC	1,DE	
4851	07647	043670	OCT	43670	DEG;111000
4852	07650	043522	ASC	2,GRAD	
	07651	040504			
4853	07652	134527	OCT	134527	GRAD;111001
4854	07653	051111	ASC	2,RITE	
	07654	052105			
4855	07655	175000	OCT	175000	WRITE;111010

PAGE 105

SYNTAX

4857* FETCH A STRING CONSTANT
4858*
4859* START STRING IN NEXT WORK OF SYNTAX BUFFER, THEN RECORD
4860* OPENING QUOTE AND THE STRING LENGTH. EXIT WITH NEXT CHAR IN (A)
4861* EXIT TO ERROR IF NO OPENING QUOTE
4862* EXIT TO ERROR IF NO CLOSING QUOTE
4863* ENTER AT GETST+3, IF OPENING QUOTE ALREADY FOUND
4864*
4865 07656 010043 GETST CPA B42 OPENING QUOTE?
4866 07657 067661 JMP *+2 YES
4867 07660 160206 NOST JSM ERRA,I NO
4868 07661 025706 LDB SBPTR SAVE INITIAL SYNTAX
4869 07662 035470 STB L2T1 BUFFER POINTER
4870 07663 074742 SBR 16 SET TO START STORING IN LEFT HALF
4871 07664 063672 JSM CHRST GET STRING
4872 07665 160206 SYE14 JSM ERRA,I NO CLOSING QUOTE
4873 07666 020116 LDA QTOP STORE OPENING QUOTE
4874 07667 001474 ADA TEMP1 AND THE
4875 07670 131470 STA L2T1,I STRING LENGTH
4876 07671 065055 JMP GNEXT GET NEXT CHAR AND RETURN
4877*
4878* PROCESS A CHARACTER STRING
4879*
4880* (A)=STRING TERMINATOR
4881* (B)=CHARACTER POINTER; 1=START IN RIGHT HALF OF CURRENT WORD
4882* 0=START IN LEFT OF NEXT WORD
4883* EXIT: P+1 ON END OF LINE
4884* P+2 ON TERMINATOR CHAR FOUND
4885* (A)=NEXT CHARACTER
4886* TEMP1=CHAR COUNT BIASED BY (B) ON ENTRY
4887* SBPTR=NEXT WORD NOT IN STRING
4888*
4889 07672 031475 CHRST STA TEMP2 RECORD TERMINATOR CHARACTER
4890 07673 035474 CHRS1 STB TEMP1 SAVE CHARACTER FLAG AND
4891 07674 061040 JSM GETCH-1 COUNTER
4892 07675 010031 CPA EOL GET CHARACTER
4893 07676 067713 JMP CHRS3+1 EOL
4894 07677 011475 CPA TEMP2 TERMINATOR?
4895 07700 067712 JMP CHRS3 YES
4896 07701 025474 LDB TEMP1
4897 07702 074251 SLB CHRS2 SELECT WORD HALF
4898 07703 141706 IOR SBPTR,I RIGHT HALF, MERGE
4899 07704 131706 STA SBPTR,I WITH PREVIOUS CHAR
4900 07705 077330 RIB CHRS1 ADVANCE COUNT
4901 07706 067673 JMP CHRS1 JUMP ON 0 COUNT
4902 07707 045706 CHRS2 ISZ SBPTR LEFT HALF, ADVANCE POINTER
4903 07710 070404 SAL 8 POSITION CHAR
4904 07711 067704 JMP *-5
4905 07712 060365 CHRS3 JSM RET2 SET TO RETURN TO P+2
4906 07713 064456 JMP SBPUD ADVANCE POINTER
4907*
4908* NEXT STATEMENT SYNTAX
4909*
4910 07714 160246 NEXTS JSM VAROA,I FETCH SIMPLE VARIABLE
4911 07715 164322 JMP E25+2,I
4912 07716 164322 JMP E25+2,I NOT FOUND

PAGE 106

SYNTAX

4913 07717 065151 JMP EOST

4914*

4915* IF STATEMENT SYNTAX

4916*

4917 07720 160202 IFS JSM FSCA,I GET DECISION FORMULA

4918 07721 027726 LDB THENA DEMAND "THEN"

4919 07722 060546 JSM STEXP AND STORE AS EXPANDED OPERATOR

4920 07723 160206 SYNE7 JSM ERRA,I NOT FOUND

4921 07724 060664 JSM LNUM DEMAND A SEQUENCE NUMBER

4922 07725 065151 JMP EOST

4923 07726 006750 THENA DEF FOP13+FOP13

4924*

4925* INITIALIZE THE TRIG UNITS

4926*

4927 07727 024012 ERAD LDB .1

4928 07730 035434 STB UNITS

4929 07731 164172 JMP XEC4A,I

4930 07732 074742 EDEG SBR 16

4931 07733 067730 JMP ERAD+1

4932 07734 024017 EGRAD LDB .2

4933 07735 067730 JMP ERAD+1

4934*

4935* FLOAT AND FIXED SYSTEM COMMAND EXECUTION

4936* (A)=0

4937*

4938 07736 072051 FLT SLA *+1,S SET TO NON ZERO FOR FLOAT

4939 07737 031451 FIXED STA CTYPE SET TO ZERO FOR FIXED

4940 07740 061066 JSM GETSK PARAMETER GIVEN?

4941 07741 164174 JMP RDYDA,I NO

4942 07742 060750 JSM DIGCK DIGIT?

4943 07743 067740 JMP FIXED+1 NO, IGNORE IT

4944 07744 055400 DSZ BADDR YES, RESET POINTER

4945 07745 024143 LDB M12

4946 07746 061131 JSM PGIN0 GET DIGIT FIELD, ALLOW 0

4947 07747 035452 STB CPLAC

4948 07750 164174 JMP RDYDA,I

4949*

4950* TRACE SYSTEM COMMAND EXECUTION

4951*

4952 07751 024062 TRAC2 LDB FLGBT

4953 07752 035441 STB TFLG1+1 SET — (MAXIMUM UPPER LIMIT)

4954 07753 076330 RIB TRAC3 SET MAXIMUM (UPPER—LOWER LIMIT)

4955 07754 071650 TRACE SZA TRAC2 ANY PARAMETERS?

4956 07755 060730 JSM GETAD FIND ADDRESSES

4957 07756 074076 TCB

4958* NORMAL: (B)=0; CLEAR TRACE FLAG

4959 07757 035441 NORML STB TFLG1+1 STORE — (UPPER LIMIT)

4960 07760 005477 ADB TEMP4

4961 07761 074076 TRAC3 TCB

4962 07762 035440 STB TFLG1 STORE (UPPER LIMIT—LOWER LIMIT)

4963 07763 164174 JMP RDYDA,I

4964*

4965*

4966* EXECUTE READ

4967*

4968 07764 160201 EREAD JSM FORMA,I GET VARIABLE ADDRESS IN HSTPT

PAGE 107

SYNTAX

4969 07765 160256
4970 07766 060412
4971 07767 021421
4972 07770 011472
4973 07771 164172
4974 07772 067764

JSM FDAAA, I FIND DATA AND SAVE IN AR2
JSM RESULT STORE DATA
LDA PRADD
CPA TEMPS MORE VARIABLES TO COME?
JMP XEC4A, I NO; RETURN
JMP EREAD YES; LOOP

PAGE 108

9830 — PROGRAM EXECUTION

4976*
 4977 10000 ORG 10000B
 4978*
 4979* INITIALIZATION
 4980*
 4981 10000 020112 XEC LDA RSTKA
 4982 10001 031777 STA RSPTR
 4983 10002 060653 JSM INITS+2
 4984 10003 062162 JSM ESTAC
 4985 10004 031416 STA MODE
 4986 10005 021725 XEC3 LDA NXTST
 4987 10006 061075 JSM FNDPS
 4988 10007 066066 JMP EEND+1
 4989 10010 066011 JMP *+1
 4990 10011 035421 ERTN1 STB PRADD
 4991* PROCEED TO XEC4
 4992*
 4993* EXECUTE NEXT PROGRAM STATEMENT
 4994*
 4995 10012 024062 XEC4 LDB FLGBT
 4996 10013 021443 LDA KEYFG
 4997 10014 050104 AND B177
 4998 10015 035443 STB KEYFG
 4999 10016 000145 ADA M14
 5000 10017 072210 RZA *+4
 5001 10020 020066 LDA INF
 5002 10021 031440 STA TFLG1
 5003 10022 066025 JMP XEC7
 5004 10023 074742 SBR 16
 5005 10024 010033 CPA B20
 5006 10025 035441 XEC7 STB TFLG1+1
 5007 10026 025416 LDB MODE
 5008 10027 074110 SZB *+2
 5009 10030 066070 JMP CALEX
 5010 10031 025442 LDB STPFL
 5011 10032 076150 RZB XEC5
 5012 10033 025421 LDB PRADD
 5013 10034 015436 CPB HFLG1
 5014 10035 066151 XEC5 JMP STOPE
 5015 10036 015437 CPB HFLG1+1
 5016 10037 066151 JMP STOPE
 5017 10040 015423 XEC6 CPB PBPTR
 5018 10041 160206 MER11 JSM ERRA1
 5019 10042 005441 ADB TFLG1+1
 5020 10043 074313 SBP XEC8
 5021 10044 005440 ADB TFLG1
 5022 10045 074212 SBM XEC8
 5023 10046 062136 JSM PLINE
 5024 10047 061337 JSM NDWRT
 5025 10050 061344 JSM CRFL1
 5026 10051 060572 XEC8 JSM NEXTL-2
 5027 10052 004135 XEC9 ADB M4
 5028 10053 005466 ADB LOCL1
 5029 10054 074437 ADB B,1
 5030 10055 074737 JMP B,1
 5031*

GUARANTEE EMPTY STACK
 RESET STACK POINTERS
 START NEW STACKS
 SET TO PROG (0)
 GET STARTING STATEMENT NUMBER
 FIND STARTING ADDRESS
 BEYOND END OF PROG
 BETWEEN LINES

MASK OFF SHIFT BIT
 RESET KEY FLAG

TRACE KEY?
 YES, SET ALL TRACE

NORMAL KEY?
 YES, CLEAR TRACE

KEYBOARD EXECUTION?
 YES, EXIT

SKIP IF STOP, STEP, OR ERROR

HALT IF ADDRESS=FLAG
 PAST END OF PROGRAM?
 YES, IMPROPER TERMINATION
 ADD —(UPPER LIMIT)
 SKIP IF NOT IN TRACE LIMITS
 ADD (UPPER LIMIT —(LOWER LIMIT))

OUTPUT LINE # TO BUFFER
 PRINT ON PRINTER

FIND OP-CODE AND PAGE
 GET JUMP TABLE STARTING ADDRESS
 ADD IN —(OP CODE)
 ADD RELATIVE JUMP ADDRESS
 JUMP TO STATEMENT EXECUTION

PAGE 109

9830 — PROGRAM EXECUTION

5032* EXECUTE LET OR IMPLIED LET STATEMENT

5033*

5034	10056	026063	ELET	LDB	DISPA	GET JUMP ADDRESS FOR DISP EXEC
5035	10057	021416		LDA	MODE	GET CALC/PROG FLAG
5036	10060	073610		RZA	ELET-2	SKIP IF CALC MODE
5037	10061	062563		JSM	FORMX	PROG—EVALUATE FORMULA
5038	10062	066012		JMP	XEC4	
5039	10063	013747	DISPA	DEF	DISP	

5040*

5041*****

5042*

5043	10064	000000	BSS	1	CHECKSUM WORD
------	-------	--------	-----	---	---------------

5044*

5045*****

5046*

5047*

5048* END EXECUTION

5049*

5050	10065	061225	EEND	JSM	LDEF	
5051	10066	060653		JSM	INITS+2	RESET POINTERS AND STACKS
5052	10067	066106		JMP	ESTOP	

5053*

5054* EXIT FOR CALC MODE

5055*

5056	10070	021421	CALEX	LDA	PRADD	
5057	10071	071652		SAM	EEND+1	SKIP IF DECOMPILED
5058	10072	000020		ADA	3	
5059	10073	031412		STA	LSTPT	RESET STACK POINTER
5060	10074	060567		JSM	UNSTK	
5061	10075	060566		JSM	UNSTK-1	RESTORE PROGRAM POINTER
5062	10076	031411		STA	TSTPT	RESTORE T-STACK POINTER
5063	10077	060567		JSM	UNSTK	
5064	10100	031410		STA	LSTAK	RESTORE
5065	10101	060567		JSM	UNSTK	
5066	10102	031412		STA	LSTPT	L-STACK POINTERS
5067	10103	121413		LDA	HSTPT,1	
5068	10104	045413		ISZ	HSTPT	
5069	10105	071713		SAP	*-2	
5070	10106		ESTOP	EQU	*	
5071	10106	160264	CALX2	JSM	CRCKA,1	YES, PROVIDE CR—LF IF NEEDED
5072	10107	070742		SAR	16	
5073	10110	031430		JMP	LNINC	GET OUT OF AUTO
5074	10111	164173		JMP	PEXMA,1	

5075*

5076* INITIALIZATION FOR KEYBOARD EXECUTION

5077*

5078	10112	025412	ECAL	LDB	LSTPT	
5079	10113	035472		STB	TEMPS	SAVE ORIGINAL STACK POINTER
5080	10114	020233		LDA	SBUFA	
5081	10115	011706	ECAL1	CPA	SBPTR	TRANSFER SYNTAX
5082	10116	066122		JMP	ECAL2	BUFFER INTO
5083	10117	070537		LDB	A,1	STACK AREA
5084	10120	060563		JSM	SLWST	
5085	10121	073630		RIA	ECAL1	
5086	10122	025472	ECAL2	LDB	TEMPS	
5087	10123	060563		JSM	SLWST	STACK OLD LSTPT

PAGE 110 9830 — PROGRAM EXECUTION

5088	10124	021412		LDA	LSTPT	GET NEW END OF LINE POINTER
5089	10125	025410		LDB	LSTAK	
5090	10126	060563		JSM	SLWST	STACK LSTPT
5091	10127	025411		LDB	TSTPT	
5092	10130	060563		JSM	SLWST	STACK TSTPT
5093	10131	060562		JSM	SLWST-1	STACK OLD PRADD
5094	10132	031421		STA	PRADD	SET PRADD TO END OF BUFFER
5095	10133	062162		JSM	ESTAC	SET UP NEW STACKS
5096	10134	060602		JSM	NEXT1-4	GET PAGE AND OP CODE
5097	10135	066052		JMP	XEC9	
5098*						
5099*						PRINT CURRENT LINE NUMBER FOR TRACE AND STOP
5100*						
5101	10136	160264	PLINE	JSM	CRCKA,I	GIVE CRLF IF PRINT
5102	10137	160170		JSM	CLERA,I	CLEAR BUFFER
5103	10140	025421		LDB	PRADD	
5104	10141	015423		CPB	PBPTR	PAST END OF PROG?
5105	10142	066146		JMP	*+4	YES, DONT PRINT LINE #
5106	10143	125421		LDB	PRADD,I	
5107	10144	074073		SBP	*+1,C	
5108	10145	160210		JSM	OUTIA,I	OUTPUT CURRENT LINE #
5109	10146	020053		LDA	COLON	
5110	10147	061016		JSM	ONEXT	OUTPUT A :
5111	10150	064373		JMP	ICNT	COMPUTE CHAR COUNT
5112*						
5113*						STOP EXECUTION
5114*						
5115	10151	074412	STOPE	SB,M	STPI	SKIP IF ERROR MESSAGE
5116	10152	062136		JSM	PLINE	OUTPUT LINE # AND COMPUTE COUNT
5117	10153	031466		STA	LOCL1	SAVE COUNT
5118	10154	021450		LDA	PMODE	
5119	10155	072210		RZA	STPI	PRINT IF PRINT-ALL
5120	10156	035701		STB	NDISP	
5121	10157	061332		JSM	NPWR+2	
5122	10160	061344		JSM	CRLF1	
5123	10161	066106	STPI	JMP	ESTOP	RETURN TO MONITOR
5124*						
5125*						
5126*						INITIALIZE STACKS FOR EXECUTION
5127*						
5128	10162	025412	ESTAC	LDB	LSTPT	START AT OLD LSTPT
5129	10163	004134		ADB	M3	
5130	10164	035411		STB	TSTPT	OFFSET T-STACK POINTER BY 4
5131	10165	004033		ADB	16	
5132	10166	035410		STB	LSTAK	ALLOW 12 WORDS FOR T-STACK
5133	10167	035412		STB	LSTPT	
5134	10170	060553		JSM	BHSTP	
5135	10171	074742		SBR	16	
5136	10172	135410		STB	LSTAK,I	
5137	10173	035432		STB	DRQST	
5138	10174	024132		LDB	M1	
5139	10175	135413		STB	HSTPT,I	
5140	10176	164170		JMP	CLERA,I	INITIALIZE THE PRINT BUFFER
5141*						
5142*						EXECUTE IF STATEMENT
5143*						

PAGE 111

9830— PROGRAM EXECUTION

5144	10177	061217	EIF	JSM	FETCH	EVALUATE FORMULA AND RETURN RESULT
5145	10200	021755		LDA	AR2+1	GET FIRST WORD OF MANTISSA
5146	10201	070250		SZA	EGOS4	SKIP IF FALSE
5147	10202	062217	EGOS1	JSM	EADD	GET ADDRESS
5148	10203	025416		LDB	MODE	
5149	10204	076150		RZB	EGOS4+1	SKIP IF KEYBOARD MODE
5150	10205	031421		STA	PRADD	STORE IN PROGRAM POINTER
5151	10206	066012	EGOS4	JMP	XEC4	
5152	10207	025421		LDB	PRADD	
5153	10210	004020		ADB	3	GET ADDRESS OF OLD PRADD
5154	10211	035466		STB	LOCL1	
5155	10212	125466		LDB	LOCL1,I	GET OLD PRADD
5156	10213	074113		SBP	*+2	
5157	10214	072053		SAP	*+1,S	SAVE DECOMPILE FLAG
5158	10215	131466		STA	LOCL1,I	
5159	10216	066012		JMP	XEC4	
5160*						
5161*						
5162*						
5163	10217	045472	EADD	ISZ	TEMPS	
5164	10220	121472		LDA	TEMPS,I	GET LINE # OR ADDRESS
5165	10221	070372		SAM	EADD1,C	ADDRESS FORMAT?
5166	10222	061075		JSM	FNDPS	NO, GET ADDRESS
5167	10223	066224		JMP	*+1	
5168	10224	160206	NGOTO	JSM	ERRA,I	NOT FOUND
5169	10225	074117		LDA	B	MOVE ADDRESS TO (A)
5170	10226	076053		SBP	*+1,S	
5171	10227	135472		STB	TEMPS,I	STORE COMPILED FORMAT
5172	10230	170402	EADD1	RET		ADDRESS IN (A)
5173*						
5174*						
5175*						
5176	10231	160206	ER3	JSM	ERRA,I	INCORRECT RETURN STATEMENT
5177	10232	125412	ERTRN	LDB	LSTPT,I	GET TOP OF RETURN STACK
5178	10233	055412		DSZ	LSTPT	
5179	10234	075650		SZB	ER3	SKIP IF NO RETURN
5180	10235	021472		LDA	TEMPS	
5181	10236	070070		SIA	*+1	
5182	10237	011421		CPA	PRADD	RETURN A FORMULA?
5183	10240	066243		JMP	*+3	NO
5184	10241	075413		SBP	ER3	YES, FUNCTION RETURN ALLOWED?
5185	10242	067106		JMP	RETN	YES, EXECUTE
5186	10243	060771		JSM	GTOPP	GET NEXT OPERAND
5187	10244	073650		RZA	*-3	RETURN AND SINGLE VARIABLE?
5188	10245	075212		SBM	ER3	NO, GOSUB RETURN ALLOWED?
5189	10246	066011		JMP	ERTN1	YES, SET RETURN ADDRESS
5190*						
5191*						
5192*						
5193	10247	011767	NOPCD	OCT	11767	ENCODED OP CODE FOR "NEXT"
5194*						
5195	10250	060771	EFOR	JSM	GTOPP	GET FOR-VARIABLE
5196	10251	160177		JSM	SSYMA,I	FIND IT IN SYMBOL TABLE
5197	10252	160252		JSM	ENTEA,I	NOT FOUND — ENTER IT
5198	10253	035474		STB	TEMP1	SAVE ABS ADDRESS ALSO
5199	10254	021487		LDA	SYMT	

PAGE 112

9830—PROGRAM EXECUTION

5200	10255	070076	TCA		
5201	10256	070037	ADB A		COMPUTE REALTIVE ADDRESS
5202	10257	020141	LDA M10		MOVE HSTPT TO LOWER
5203	10260	001413	ADA HSTPT		ADDRESS TO ACCOMMODATE
5204	10261	031413	STA HSTPT		THE NEW FOR VAR.
5205	10262	031476	STA TEMP3		SAVE IT FOR MVTOH IF NEEDED.
5206	10263	135413	STB HSTPT,1		SAVE RELATIVE SYMBOL TABLE
					POINTER
5207*					
5208	10264	060554	JSM BHSTP+1		OVERLAY?
5209*					
5210	10265	000061	EFOR2	ADA .10	
5211	10266	070537	LDB A,1		LOAD THE REALTIVE SYMBOL TABLE
					POINTER
5212	10267	074512	SBM EFOR1		IF MINUS MEANS BOOTOM OF HIGH
					STACK
5213	10270	031475	STA TEMP2		SAVE THIS ADDRESS FOR MVTOH IF
					NEEDED
5214	10271	115413	CPB HSTPT,1		IS THIS F.V. THE SAME ?
5215	10272	066274	JMP *+2		YES — GO DELETE IT
5216	10273	066265	JMP EFOR2		NO — LOOP BACK FOR NEXT ONE
5217*					
5218	10274	000061	ADA .10		SET UP LAST WORD OF DESTINATION
5219	10275	031477	STA TEMP4		
5220	10276	061121	JSM MVTOH		DELETE IT AND MOVE REST TO HIGHER
					MEMORY
5221	10277	025477	LDB TEMP4		
5222	10300	035413	STB HSTPT		RESET HSTPT
5223*					
5224	10301	160201	EFOR1	JSM FORMA,1	
5225*					
5226	10302	061371	JSM FIX		GET (A) AND (B) FIXED UP
5227	10303	060416	JSM STORE		STORE AR2 IN INITIAL VALUE TABLE
5228*					
5229	10304	061217	JSM FETCH		GET ENDING LIMIT
5230	10305	025413	LDB HSTPT		TRANSFER END LIMIT
5231	10306	074070	SIB *+1		TO HSTPT+1 THUR +4
5232	10307	060406	JSM XFAR1+1		
5233*					
5234	10310	020020	LDA ONE		
5235	10311	060445	JSM XFAR2+1		SET AR2=1
5236	10312	021472	LDA TEMPS		IS THERE A STEP SIZE?
5237	10313	011421	CPA PRADD		
5238	10314	066316	JMP *+2		NO, LEAVE 1 IN AR2
5239	10315	061217	JSM FETCH		YES — GET IT
5240	10316	025413	LDB HSTPT		TRANSFER STEP
5241	10317	004022	ADB .5		SIZE
5242	10320	060406	JSM XFAR1+1		TO HSTPT+4
5243*					
5244	10321	004021	ADB .4		STORE NEXT
5245	10322	021421	LDA PRADD		EXECUTABLE
5246	10323	074557	STA B,1		LINE ADDRESS IN HSTPT+9
5247*					
5248*					
5249*					
5250*					
5251*					
5252*					
5253*					
5254*					
5255*					

FIND THE CORRESPONDING NEXT STATEMENT (TEMPS AND PRADD WILL BE UP-DATED ACCORDINGLY). THEN CHECK IF THE "FOR" LIMITS ARE CORRECT FOR LOOPING. IF NOT — ERASE FOR INFORMATION AND JMP TO XEC4 TO CONTINUE AFTER THE "NEXT" STATEMENT. IF THE "FOR" LOOP WAS VALID, LOAD HSTPT+9 (THE ADDRESS FOLLOWING THE "FOR" STATEMENT) AND EXECUTE FROM THERE (XEC4+1).

PAGE 113

9830—PROGRAM EXECUTION

5256	10324	022247	LDA	NOPCD	LOAD SEARCH CODE
5257	10325	062413	JSM	SYTSH	FIND A NEXT STATEMENT
5258	10326	074373	SBP	BERR,C	B IS NEGATIVE IF FOUND
5259	10327	062405	JSM	TEST	GO CHECK IF SAME VARIABLE
5260	10330	075613	SBP	*-4	LOOP BACK UNLESS FOUND
5261*					
5262	10331	061371	JSM	FIX	GET A AND B REG FIXED FOR
5263	10332	063444	JSM	OPCHI	FOR CONVERTING INT, SPLIT, OR
					FULL TO FULL
5264	10333	035475	STB	TEMP2	
5265	10334	066361	JMP	FTEST	CHECK IF LOOP IS VALID
5266*					
5267	10335	160206	BERR	JSM	ERRA,I
					NO CORRESPONDING "NEXT" OR
					INCORRECT NESTING
5268*					
5269*					
5270*					
5271*					
5272	10336	125413	ENEXT	LDB	HSTPT,I
5273	10337	075712	SBM	BERR	RECOVER SYMBOL TABLE ADDRESS
5274	10340	005407	ADB	SYMTP	ERROR BOTTOM OF STACK
5275	10341	035474	STB	TEMP1	COMPUTE ABS ADDRESS OF VARIABLE
5276	10342	062405	JSM	TEST	
5277	10343	074252	SBM	*+5	IF B IS -, GET OUT OF LOOP
5278	10344	020061	LDA	.10	IF + THE VARIABLE WASN'T FOUND
5279	10345	001413	ADA	HSTPT	SO DINC THE HSTPT
5280	10346	031413	STA	HSTPT	AND LOOP BACK
5281	10347	066336	JMP	ENEXT	
5282*					
5283*					
5284*					
5285*					
5286*					
5287*					
5288	10350	061371	JSM	FIX	FIX A AND B FOR
5289	10351	063444	JSM	OPCHI	CONVERT INT OR SPLIT TO FULL
5290	10352	021413	LDA	HSTPT	ON RETURN (B) IS POINTING AT
					NUMBER
5291	10353	000022	ADA	.5	LOAD (A) WITH STEP SIZE ADDRESS
5292	10354	160216	JSM	.F,ADA,I	
5293*					
5294	10355	063454	JSM	OPCI	SAVE AR2 AT 1470B
5295	10356	035475	STB	TEMP2	SAVE 1470B IN TEMP2
5296*					
5297	10357	061371	JSM	FIX	FIX (A) AND (B) FOR XFR
5298	10360	060416	JSM	STORE	STORE ANSWER BACK IN VALUE
					TABLE
5299*					
5300	10361	025413	FTEST	LDB	HSTPT
5301	10362	074070	SIB	*+1	(B) = ADDRESS OF END LIMIT
5302	10363	021413	LDA	HSTPT	GET SIGN
5303	10364	000022	ADA	.5	OF STEP
5304	10365	070517	LDA	A,1	SIZE
5305	10366	070151	SLA	*+3	
5306	10367	021475	LDA	TEMP2	STEP SIZE IS -, F.V.—END LIMIT = AR2
5307	10370	066373	JMP	*+3	
5308	10371	074117	LDA	B	STEP SIZE IS +, END LIMIT—F.V. = AR2
5309	10372	025475	LDB	TEMP2	
5310	10373	160217	JSM	.FSBA,I	
5311*					

221

219

1,444,141

219

PAGE 114

9830—PROGRAM EXECUTION

5312	10374	021413	LDA	HSTPT	(A) IS POINTING AT NEXT
5313	10375	000026	ADA	.9	EXECUTABLE LINE ADDRESS
5314	10376	025754	LDB	AR2	TEXT AR2 FOR + RESULT
5315	10377	074211	SLB	*+4	
5316*					
5317	10400	070070	SIA	*+1	AR2 WAS NEGATIVE, ERASE
5318	10401	031413	STA	HSTPT	"FOR" LOOP INFORMATION
5319	10402	066012	JMP	XEC4	AND EXECUTE LINE AFTER "NEXT"
5320*					
5321	10403	070537	LDB	A,I	AR2 WAS POSITIVE OR 0.
5322	10404	066011	JMP	ERTN1	EXECUTE STMT AFTER "FOR"

PAGE 115

9830—PROGRAM EXECUTION

5324*
5325* SUBROUTINES FOR "FOR" AND "NEXT" STATEMENTS.
5326*
5327 10405 061371 TEST JSM FIX RETURN WITH TYPE CODE OF F.V.
5328 10406 141472 IOR TEMPS,I IN A, IOR IN THE F.V. AND PAGE
5329 10407 050115 AND B1777 REMOVE PAGE
5330 10410 111474 CPA TEMPI,I DO THE VARIABLES AGREE?
5331 10411 076053 SBP *+1,S THEY WERE THE SAME, FLAG B
5332 10412 170402 RET DID NOT AGREE B IS 1.
5333*
5334* CALLING SEQUENCE FOR SYTSH
5335*
5336* LDA OP CODE (PAGE-OP CODE)
5337* JSM SYTSH (TEMPS IS POINTING AT LINE #.)
5338* ON RETURN
5339* (B)=-, OP CODE FOUND PRADD IS POINTING AT NEXT LINE #.
5340* TEMPS IS POINTING AT PRADD+2.
5341* (B)=+ ERROR RETURN = OP CODE NOT FOUND
5342*
5343 10413 031475 SYTSH STA TEMP2 SAVE SEARCH CODE
5344 10414 025421 LDB PRADD
5345 10415 015423 CPB PBPTR CHECK IF END OF PROGRAM
5346 10416 170402 RET YES — RETURN WITH (B) +.
5347 10417 035472 STB TEMPS NO — SET TEMPS PRADD
5348 10420 060577 JSM NEXTL+3 LOOK AT THE NEXT STATEMENT.
5349 10421 001466 ADA LOCL1 ENCODE PAGE AND OP CODE
5350 10422 011475 CPA TEMP2 WAS IT TO TEMP2(SEARCH CODE)?
5351 10423 077553 SBP *-5,S YES, RETURN WITH (B) NEGATIVE
5352 10424 066414 JMP SYTSH+1 NO — LOOP BACK.

PAGE 116

FORMULA EVALUATION ROUTINES

5354*
 5355*
 5356* FIND THE NEXT DATA ITEM AND MOVE IT TO AR2
 5357* NXTDT MUST BE POINTING AT THE NEXT DATA ITEM OR
 5358* EQUAL TO DSTRT. IF EQUAL TO DSTRT FDATA WILL SEARCH
 5359* FOR THE NEXT DATA STATEMENT.
 5360*
 5361 10425 062437 FDATA JSM SWITH SAVE TEMPS, TEMPS = NXTDT
 5362 10426 021421 LDA PRADD
 5363 10427 031642 STA SAVE
 5364 10430 025472 LDB TEMPS
 5365 10431 015414 CPB DSTRT IF NXTDT(TEMPS) AND DSTRT ARE =;
 5366 10432 066445 JMP .FDA1 GO FIND NEXT DATA STATEMENT
 5367 10433 160232 JSM SNFLA,I OR CONVERT SYNTAX TO FULL
 5368 10434 045472 ISZ TEMPS
 5369 10435 021642 LDA SAVE
 5370 10436 031421 STA PRADD
 5371 10437 021472 SWITH LDA TEMPS
 5372 10440 025415 LDB NXTDT
 5373 10441 031415 STA NXTDT
 5374 10442 035472 STB TEMPS
 5375 10443 170402 RET
 5376*
 5377 10444 160206 ER60 JSM ERRAI OUT OF DATA
 5378 10445 035421 .FDA1 STB PRADD
 5379 10446 022454 LDA DATCD LOAD "DATA" STATEMENT SEARCH
 5380 10447 062413 JSM SYTSH FIND NEXT DATA STATEMENT
 5381 10450 075613 SBP ER60 IF (B) WAS +, NO MORE DATA FOUND
 5382 10451 025421 LDB PRADD
 5383 10452 035414 STB DSTRT
 5384 10453 066430 JMP FDATA+3 DATA WAS FOUND GO CHECK IF ANY
 5385* DATA WITH IT
 5386 10454 011761 DATCD OCT 11761

PAGE 117

FORMULA EVALUATION ROUTINES

```

5388*
5389* EXECUTE INPUT *
5390*
5391 10455 021622 EINPT LDA TERM
5392 10456 072110 RZA *+2
5393 10457 160170 JSM CLERA,I
5394 10460 021472 LDA TEMPS
5395 10461 000133 ADA M2
5396 10462 031641 STA SPRDD SAVE START OF INPUT STATEMENT
5397 10463 021641 LDA SPRDD SET PRADD TO START OF
5398 10464 031421 STA PRADD INPUT STATEMENT
5399 10465 020262 LDA INREA SET UP RETURN
5400 10466 031432 STA DRQST SET DATA REQUEST FLAG
5401 10467 164173 INRET JMP PEXMA,I CALL FOR DATA
5402 10470 055400 DSZ BADDR
5403 10471 025641 LDB SPRDD
5404 10472 061007 JSM NEXTA SET PRADD BACK TO END
5405 10473 035421 STB PRADD OF INPUT STATEMENT
5406 10474 021472 LDA TEMPS
5407 10475 031640 STA TEPPI SAVE START OF VARIABLE
5408 10476 160201 JSM FORMA,I GET THE ADDRESS OF THE VARIABLE
IN HSTPT

5409 10477 024017 LDB 2
5410 10500 035706 STB SBPTR AIM SBPTR AT ROM
5411 10501 026532 LDB INCHA
5412 10502 035432 STB DRQST SET FLAG IN CASE OF ERROR
5413 10503 061156 JSM CONST BUILD A NUMBER IN AR2
5414 10504 066520 JMP INCHT-1 NO NUMBER FOUND, RESTORE
H-STACK

5415 10505 070742 SAR 16
5416 10506 031432 STA DRQST CLEAR FLAG
5417 10507 060412 JSM RESULT SAVE AR2 AT HSTPT,I
5418 10510 025472 LDB TEMPS
5419 10511 015421 CPB PRADD END OF STATEMENT?
5420 10512 066530 JMP CXEC4 YES
5421 10513 035640 STB TEPPI RESET THE TEMPORARY POINTER
5422 10514 055400 DSZ BADDR NO — DINC BADDR TO LOOK AT THE
LAST CHAR

5423 10515 061066 JSM GETSK GET LAST CHAR.
5424 10516 066526 JMP INCH1
5425 10517 066476 JMP EIN1 P+2 RETURN — OTHER CHAR'S

5426*
5427 10520 160170 JSM CLERA,I
5428 10521 021640 INCHT LDA TEPPI
5429 10522 031472 STA TEMPS RESTORE VARIABLE POINTER
5430 10523 045413 ISZ HSTPT
5431 10524 045413 ISZ HSTPT
5432 10525 066463 JMP INRET-4

5433*
5434 10526 160170 INCH1 JSM CLERA,I
5435 10527 066463 JMP INRET-4

5436*
5437 10530 160170 CXEC4 JSM CLERA,I
5438 10531 066012 JMP XEC4
5439*
5440 10532 010521 INCHA DEF INCHT
5441* EXECUTE RESTORE STATEMENT
5442*
5443*

```

234

PAGE 118

FORMULA EVALUATION ROUTINES

5444 10533 125472 ERSTR
5445 10534 021424
5446 10535 074113
5447 10536 062217
5448 10537 031415
5449 10540 031414
5450 10541 066012

LDB TEMPS,I
LDA PBUFF
SBP *+2
JSM EADD
STA NXTDT
STA DSTRT
JMP XEC4

LINE GIVEN?
YES, GET ADDRESS
SET UP DATA POINTERS

PAGE 119

FORMULA EVALUATION ROUTINES

```

5452*
5453* EVALUATE ARGUMENT OF FUNCTION
5454*
5455 10542 024032 EVARG LDB B17 INITIALIZE OPERATOR STACK
5456 10543 066564 JMP FORMX+1 FOR FUNCTION
5457*
5458* EVALUATE A FORMULA
5459*
5460*
5461* STACK A CONSTANT
5462 10544 050040 FORM6 AND B37
5463 10545 010040 CPA B37 PRE-DEFINED FUNCTION?
5464 10546 066664 JMP FORM7 YES
5465 10547 055413 DSZ HSTPT BUMP STACK POINTER
5466 10550 063406 JSM RSCHK ALLOT SPACE ON T-STACK
5467 10551 160232 JSM SNFLA,I CONVERT TO FLOATING
5468 10552 045472 FORST ISZ TEMPS
5469 10553 074742 SBR 16
5470 10554 135413 STB HSTPT,I STORE FULL PRECISION TYPE
5471 10555 060553 JSM BHSTP
5472 10556 025411 LDB TSTPT
5473 10557 135413 STB HSTPT,I STORE ADDRESS OF T-STACK
5474 10560 020214 LDA AR2A
5475 10561 170004 XFR TRANSFER RESULT TO T-STACK
5476 10562 066616 FOR2L JMP FORM2
5477*
5478*
5479 10563 074742 FORMAX SBR 16 INITIALIZE
5480 10564 060563 JSM SLWST OPERATOR STACK
5481* PROCESS NEXT OPERAND
5482 10565 121472 FORM1 LDA TEMPS,I FETCH OPERAND
5483 10566 050070 AND OPDMK
5484 10567 070113 SAP *+2
5485 10570 066544 JMP FORM6 CONSTANT OR FUNCTION
5486 10571 045472 ISZ TEMPS SET FOR NEXT WORD OF FORMULA
5487 10572 071410 SZA FOR2L SKIP IF NULL OPERAND
5488 10573 055413 DSZ HSTPT BUMP POINTER
5489 10574 025420 LDB PARMN GET PARAMETER NAME
5490 10575 074504 SBL 6
5491 10576 074242 SBR 6 MASK OFF PRECISION
5492 10577 074057 CPA B IS THIS OPERAND THE PARMETER
5493 10600 067116 JMP FPARM YES
5494* STACK NON-FUNCTION VARIABLE OPERAND ADDRESS
5495 10601 160177 JSM SSYMA,I SEARCH FOR VARIABLE IN SYMTAB
5496 10602 160252 JSM ENTEA,I NOT FOUND, ENTER IT
5497 10603 074517 LDA B,I
5498 10604 050040 AND B37 FETCH TYPE
5499 10605 004132 ADB M1
5500 10606 035750 STB YC SAVE ADDRESS-ADDRESS
5501 10607 125750 LDB YC,I READ ADDRESS
5502 10610 010040 CPA B37
5503 10611 066725 JMP FORM8 IT WAS A USER DEFINED FUNCTION
5504 10612 131413 FORMP STA HSTPT,I SAVE TYPE ON H-STACK
5505 10613 074117 LDA B GET ADDRESS OF OPERAND
5506 10614 060553 JSM BHSTP BUMP POINTER
5507 10615 131413 STA HSTPT,I STORE ADDRESS ON OPERAND STACK

```


PAGE 120

FORMULA EVALUATION ROUTINES

```

5508*  PROCESS NEXT OPERATOR
5509  10616 021472 FORM2 LDA TEMPS  END OF
5510  10617 011421 CPA PRADD  STATEMENT?
5511  10620 066631 JMP FORM3 YES, FORCE END OF FORMULA
5512  10621 121472 LDA TEMPS,1 EXTRACT
5513  10622 070073 SAP *+1,C
5514  10623 070442 SAR 10 NEXT
5515  10624 070137 LDB A OPERATOR
5516  10625 000135 ADA M4 OP CODE LESS THAN 4?
5517  10626 070152 SAM FORM3 IF YES, NON FORMULA OPERATOR
5518  10627 000147 ADA M23 NO, GREATER THAN B32?
5519  10630 070112 SAM *+2 NON-FORMULA OPERATOR?
5520  10631 024012 FORM3 LDB 1 YES, USE CODE FOR QUOTE
5521  10632 035460 STB FTMP1 SAVE CURRENT OP-CODE
5522  10633 004123 ADB FOPBS ADD OP CODE TO TABLE ADDRESS
5523  10634 074517 LDA B,1 GET AND MASK
5524  10635 050040 AND B37 PRECEDENCE
5525  10636 031461 STA FTMP2 SAVE PRECEDENCE OF CURRENT OP
5526*  STACK PRESENT OR EXECUTE PREVIOUS OPERATOR
5527  10637 121412 FORM4 LDA LSTPT,1 DOES OPERATOR ON
5528  10640 050040 AND B37 TOP OF OPERATOR
5529  10641 070056 CMA STACK HAVE
5530  10642 001461 ADA FTMP2 HIGHER
5531  10643 070412 SAM FORM5 PRECEDENCE?
5532  10644 025461 FORM9 LDB FTMP2 NO, STACK CURRENT OPERATOR
5533  10645 014033 CPB B20 LEFT PAREN OR BRACKET?
5534  10646 024012 LDB 1 YES, STACK PRECEDENCE OF 1
5535  10647 021460 LDA FTMP1 COMBINE
5536  10650 070304 SAL 10 PRECEDENCE
5537  10651 070037 ADB A WITH OP-CODE
5538  10652 066564 JMP FORM1-1 AND STACK IT
5539  10653 121412 FORM5 LDA LSTPT,1
5540  10654 055412 DSZ LSTPT
5541  10655 070442 SAR 10 COMPUTE JUMP ADDRESS
5542  10656 003135 ADA ARBAS FOR OPERATOR
5543*  AT THIS POINT (HSTPT,1)= ADDRESS OF THE TOP OPERAND
5544*  (HSTPT+1,1)= THE TYPE OF THE TOP OPERAND
5545  10657 070737 JMP A,1 BRANCH TO ROUTINE
5546*
5547*  EVALUATE NON-ARGUMENT FUNCTION
5548  10660 055413 FOR72 DSZ HSTPT
5549  10661 060553 JSM BHSTP PLACE DUMMY ON STACK
5550  10662 045472 ISZ TEMPS POINT TO NEXT OPERATOR
5551  10663 066707 JMP FOR71
5552*  EVALUATE PRE-DEFINED FUNCTION
5553  10664 025777 FORM7 LDB RSPTR
5554  10665 004132 ADB M1 TAKE RETURN ADDRESS OF
5555  10666 074537 LDB B,1 FORMX OFF STACK AND
5556  10667 055777 DSZ RSPTR
5557  10670 060563 JSM SLWST SAVE ON LOW STACK
5558  10671 125472 LDB TEMPS,1 GET OP CODE
5559  10672 074544 SBL 5
5560  10673 060604 JSM NEXT1-2 CONFIGURE OP CODE AND GET PAGE
5561  10674 004134 ADB M3
5562  10675 074437 ADB B,1 GET TABLE STARTING ADDRESS
5563  10676 005466 ADB LOCL1 ADD -(OP CODE)

```

PAGE 121

FORMULA EVALUATION ROUTINES

5564	10677	074437	ADB B,I	GET ACTUAL JUMP ADDRESS
5565	10700	060563	JSM SLWST	STACK JUMP ADDRESS
5566	10701	021466	LDA LOCL	
5567	10702	000036	ADA .23	
5568	10703	070113	SAP *+2	ARGUMENT REQUIRED?
5569	10704	066660	JMP FOR72	NO
5570	10705	062542	JSM EVARG	EVALUATE ARGUMENT
5571	10706	063433	JSM OPCHK	VERIFY RESULT
5572	10707	063405	JSM RSCHK-1	ALLOT SPACE ON T-STACK
5573	10710	125412	LDB LSTPT,I	
5574	10711	055412	DSZ LSTPT	REMOVE ADDRESS FROM STACK
5575	10712	074717	JSM B,I	BRANCH TO ROUTINE
5576*	RECORD RESULT OF FUNCTION			
5577*				
5578*	FUNCTIONS RETURN HERE WITH RESULT IN AR2			
5579	10713	045413	FOR10 ISZ HSTPT	POP ARGUMENT BUT LEAVE STACK BUMPED!
5580	10714	121412	LDA LSTPT,I	UNSTACK RETURN ADDRESS
5581	10715	055412	DSZ LSTPT	
5582	10716	025777	LDB RSPTR	GET RETURN ADDRESS
5583	10717	045777	ISZ RSPTR	
5584	10720	070173	SAP *+3,C	MORE ADDRESSES TO COME?
5585	10721	074557	STA B,I	YES
5586	10722	066714	JMP FOR10+1	GET NEXT RETURN ADDRESS
5587	10723	074557	STA B,I	NO
5588	10724	066553	JMP FORST+1	
5589*	EVALUATE PROGRAMMER DEFINED FUNCTION			
5590	10725	076110	FORM8 RZB *+2	ADDRESS=0?
5591	10726	160222	JSM ENTFA,I	YES, FIND FUNCTION
5592	10727	074742	SBR 16	FLAG FIRST RETURN, SIGN=0
5593	10730	021777	FMX80 LDA RSPTR	
5594	10731	000132	ADA M1	
5595	10732	070437	ADB A,I	COMBINE RETURN ADDRESS WITH FLAG
5596	10733	055777	DSZ RSPTR	
5597	10734	060563	JSM SLWST	STACK ON L-STACK
5598	10735	021777	LDA RSPTR	
5599	10736	010112	CPA RSTKA	ALL OF STACK MOVED?
5600	10737	066742	JMP *+3	YES
5601	10740	024062	LDB FLGBT	
5602	10741	066730	JMP FMX80	
5603	10742	020132	LDA M1	
5604	10743	131413	STA HSTPT,I	FLAG STACK FOR "FOR" LOOPS
5605	10744	025411	LDB TSTPT	
5606	10745	060563	JSM SLWST	SAVE T-STACK POINTER
5607	10746	125750	LDB YC,I	
5608	10747	060563	JSM SLWST	SAVE FUNCTION ADDRESS
5609	10750	062542	JSM EVARG	EVALUATE ARGUMENT
5610	10751	021472	LDA TEMPS	SWITCH FORMULA
5611	10752	125412	LDB LSTPT,I	POINTER TO FUNCTION FORMULA
5612	10753	035472	STB TEMPS	
5613	10754	131412	STA LSTPT,I	
5614	10755	023100	LDA DSTRA	MOVE LSTPT UP 9 WORDS
5615	10756	024026	LDB .9	AND TRANSFER DSTR
5616	10757	063122	JSM MOVST	THROUGH PBUFF
5617	10760	170004	XFR	TO L-STACK
5618	10761	121413	LDA HSTPT,I	GET PARAMETER ADDRESS
5619	10762	031417	STA PARMA	



PAGE 122

FORMULA EVALUATION ROUTINES

5620	10763	011411	CPA	TSTPT	IS PARAMETER ON T-STACK?
5621	10764	063406	JSM	RSCHK	YES, PROTECT IT
5622	10765	025423	LDG	PBPTR	
5623	10766	074076	TCB		
5624	10767	005472	ADB	TEMPS	
5625	10770	074313	SBP	FMX82	SKIP IF OUTSIDE PRESENT PROGRAM
5626	10771	025424	LDB	PBUFF	
5627	10772	074076	TCB		
5628	10773	005472	ADB	TEMPS	
5629	10774	074112	SBM	FMX82	SKIP IF OUTSIDE PRESENT PROGRAM
5630	10775	067016	JMP	FMX81	
5631	10776	061225	JSM	LDEF	INSURE NOT IN KEY MODE
5632	10777	024166	LDB	FWAM	FIND PROGRAM CONTAINING
					FUNCTION
5633	11000	015405	FMX84	CPB	FWUP
5634	11001	067013	JMP	FMX83	END OF KEYS?
5635	11002	074070	SIB	*+1	YES, MUST BE MAINLINE
5636	11003	035474	SIB	TEMP1	POINT TO LENGTH WORD
5637	11004	074437	ADB	B,I	SAVE START
5638	11005	074117	LDA	B	COMPUTE END
5639	11006	070076	TCA		
5640	11007	001427	ADA	TEMPS	
5641	11010	071413	SAP	FMX84	FUNCTION IN THIS KEY?
5642	11011	025474	LDB	TEMP1	YES
5643	11012	160263	JSM	SUMPA,I	YES, SET UP PROGRAM POINTERS
5644	11013	025424	FMX83	LDB	PBUFF
5645	11014	035414	STB	DSTRT	RESET DATA STMT POINTERS
5646	11015	035415	STB	NXTDT	
5647	11016	070742	FMX81	SAR	16
5648	11017	031416	STA	MODE	SET TO PROGRAM MODE
5649	11020	025472	LDB	TEMPS	
5650	11021	061007	JSM	NEXTA	
5651	11022	035421	STB	PRADD	SET POINTER TO END OF LINE
5652	11023	045413	ISZ	HSTPT	
5653	11024	125413	LDB	HSTPT,I	
5654	11025	074244	SBL	11	GET PRECISION OF PARAMETER
5655	11026	045413	ISZ	HSTPT	DINK PARAMETER OFF STACK
5656	11027	021472	LDA	TEMPS	
5657	11030	000020	ADA	3	
5658	11031	070517	LDA	A,I	FETCH PARAMETER NAME
5659	11032	050115	AND	B1777	
5660	11033	074217	IOR	B	MERGE WITH PRECISION
5661	11034	031420	STA	PARMN	
5662	11035	025472	LDB	TEMPS	
5663	11036	004022	ADB	5	POINT PAST PARAMETER
5664	11037	015421	CPB	PRADD	MULTI-LINE FUNCTION?
5665	11040	067102	JMP	EVMLF	YES
5666	11041	035472	STB	TEMPS	NO
5667*					
5668	11042	062563	FMX9	JSM	FORMX
5669	11043	121412	LDA	LSTPT,I	EVALUATE FUNCTION FORMULA
5670	11044	027076	LDB	PBUFA	FETCH OLD PBUFF
5671	11045	035477	STB	TEMP4	
5672	11046	025412	LDB	LSTPT	INITIALIZE TRANSFER POINTER
5673	11047	074070	SIB	*+1	
5674	11050	035475	STB	TEMP2	
5675	11051	004141	ADB	M10	

PAGE 123

FORMULA EVALUATION ROUTINES

5676	11052	035412	STB	LSTPT	RESET STACK POINTER
5677	11053	074070	SIB	*+1	
5678	11054	011424	CPA	PBUFF	WAS A KEY ENTERED?
5679	11055	004017	ADB	.2	NO, DONT RESTORE DATA POINTERS
5680	11056	035476	STB	TEMP3	
5681	11057	061121	JSM	MVTOH	RESTORE PROGRAM POINTERS
5682	11060	121412	LDA	LSTPT,I	
5683	11061	055412	DSZ	LSTPT	
5684	11062	031472	STA	TEMPS	RESTORE PROGRAM POINTER
5685	11063	121412	LDA	LSTPT,I	
5686	11064	055412	DSZ	LSTPT	
5687	11065	031411	STA	TSTPT	RESTORE T-STACK POINTER
5688	11066	063404	JSM	STTOP	GET ADDRESS OF RESULT
5689	11067	024214	LDB	AR2A	
5690	11070	170004	XFR		TRANSFER
5691	11071	121413	LDA	HSTPT,I	REMOVE ANY UNTERMINATED
5692	11072	010132	CPA	M1	"FOR" INFORMATION
5693	11073	066714	JMP	FOR10+1	
5694	11074	045413	ISZ	HSTPT	
5695	11075	067071	JMP	*+4	
5696	11076	001425	PBUFA	DEF PBUFF+1	
5697	11077	177771	M7	DEC -7	
5698	11100	001414	DSTRA	DEF DSTRT	
5699	11101	001640	STADD	DEF ST	
5700*					
5701*					EVALUATE MULTI LINE FUNCTION
5702	11102	023101	EVMLF	LDA STADD	TRANSFER FIRST 7 WORDS
5703	11103	024025		LDB .8	OF EXECUTION TEMPORARY AREA
5704	11104	063122		JSM MOVST	TO L-STACK
5705	11105	066012		JMP XEC4	
5706*					
5707	11106	021412	RETNN	LDA LSTPT	
5708	11107	003077		ADA M7	RESTORE EXECUTION
5709	11110	031412		STA LSTPT	
5710	11111	070070		SIA *+1	
5711	11112	027101		LDB STADD	TEMPROARIES
5712	11113	170004		XFR	
5713	11114	170004		XFR	
5714	11115	067042		JMP FMX9	
5715*					
5716*					SET UP ADDRESS AND TYPE FOR PARAMETER
5717	11116	021420	FPRAM	LDA PARMN	PRECISION IS STORED IN
5718	11117	070502		SAR 11	UPPER BITS
5719	11120	025417		LDB PARMA	
5720	11121	066612		JMP FORMP	
5721*					
5722*					TRANSFER WORDS TO LOW STOCK
5723*					(A)=TRANSFER STARTING ADDRESS
5724*					(B)= # OF WORDS
5725*					
5726	11122	035467	MOVST	STB LOCL2	SAVE COUNT
5727	11123	025412		LDB LSTPT	SAVE OLD LSTPT
5728	11124	035466		STB LOCL1	
5729	11125	005467		ADB LOCL2	ADD COUNT TO POINTER
5730	11126	035412		STB LSTPT	
5731	11127	060554	JSM	BHSTP+1	CHECK FOR OVERFLOW

PAGE 124 FORMULA EVALUATION ROUTINES

5732	11130	135412	STB	LSTPT,I	STACK NEGATIVE FLAG FOR "RETURN"
5733	11131	025466	LDB	LOCL1	STMT
5734	11132	074070	SIB	*+1	
5735	11133	170004	XFR	TRANSFER 4 WORDS	
5736	11134	064446	JMP	XFAR2+2TRANSFER 3 MORE AND RETURN	

PAGE 125 FORMULA EVALUATION ROUTINES

5738*
5739* FORMULA OPERATOR JUMP TABLE
5740*
5741 11135 111136 ARBAS DEF *+1,I
5742 11136 011432 DEF FORMR EXECUTE END OF FORMULA
5743*
5744* THE FOLLOWING 3 TABLE WORDS ARE NOT USED
5745*
5746* QUOTE, COMMA, AND SEMI-COLON ARE END OF FORMULA OPERATORS
5747*
5748* CALL SUBTRACT
5749*
5750 11137 063376 EFSB JSM BINOP
5751 11140 160217 JSM FSBA,I
5752 11141 067260 JMP FOR14
5753*
5754 11142 011432 DEF FORMR ')
5755 11143 011432 DEF FORMR ']
5756 11144 011202 DEF ESCMA SUBSCRIPT COMMA
5757 11145 011171 DEF ESTR ASSIGNMENT EQUAL
5758 11146 000000 NOP ***** CHECKSUM WORD *****
5759 11147 011256 DEF EFAD
5760 11150 011137 DEF EFSB
5761 11151 011272 DEF EFMP
5762 11152 011275 DEF EFDV
5763 11153 011300 DEF EPWR
5764 11154 011333 DEF EGTRT
5765 11155 011336 DEF ELST
5766 11156 011352 DEF ENEQL
5767 11157 011344 DEF EEQL LOGICAL EQUAL
5768 11160 011303 DEF EUMIN UNARY MINUS
5769 11161 010637 DEF FORM4 UNARY PLUS
5770 11162 011317 DEF ELBRC '['
5771 11163 010565 DEF FORM1 '('
5772 11164 011357 DEF EOR
5773 11165 011366 DEF EAND
5774 11166 011716 DEF ENOT
5775 11167 011341 DEF EGORE
5776 11170 011347 DEF ELORE '<='

PAGE 126

FORMULA EVALUATION ROUTINES

5778*
5779* EXECUTE ASSIGNMENT EQUAL
5780*
5781 11171 025461 ESTR LDB FTMP2 IS NEXT OPERATOR AN
5782 11172 074150 SZB *+3 END OF FORMULA?
5783 11173 045412 ISZ LSTPT NO, MULTIPLE STORE—RESTORE
5784 11174 066644 JMP FORM9 STORE OPERATOR AND DEFER
STORE
5785 11175 055460 DSZ FTMP1 FIRST STORE EXECUTED?
5786 11176 067200 JMP ESTR1 NO SOURCE IS ALREADY IN AR2
5787 11177 061220 JSM FETCH+1 POP STACK AND MOVE SOURCE TO
AR2
5788 11200 060412 ESTR1 JSM RESULT POP DESTINATION AND STORE
5789 11201 066637 JMP FORM4
5790*
5791* EXECUTE SUBSCRIPT COMMA
5792*
5793 11202 061220 ESCMA JSM FETCH+1 CHECK COLUMN SUBSCRIPT
5794 11203 160225 JSM FLTRA,I ROUND TO INTEGER
5795 11204 067240 JMP ER6 OVERFLOW
5796 11205 004132 ADB M1 ADD OFFSET
5797 11206 075712 SBM *-2 UNDERFLOW?
5798 11207 135412 STB LSTPT,I SAVE COLUMN SUBSCRIPT
5799 11210 061220 JSM FETCH+1 CHECK ROW SUBSCRIPT
5800 11211 160225 JSM FLTRA,I ROUND TO INTEGER
5801 11212 067240 JMP ER6
5802 11213 004132 ADB M1
5803 11214 075712 SBM *-2
5804 11215 035471 STB L2T2 SAVE ROW SUBSCRIPT
5805 11216 125413 LDB HSTPT,I GET ACTUAL BOUNDS
5806 11217 004132 ADB M1
5807 11220 074517 LDA B,I
5808 11221 050105 AND B377 EXTRACT COLUMN BOUND
5809 11222 070070 SIA *+1
5810 11223 031470 STA L2T1 SAVE IT
5811 11224 074537 LDB B,I EXTRACT ROW BOUND
5812 11225 074342 SBR 8
5813 11226 074070 SIB *+1
5814 11227 074076 TCB
5815 11230 005471 ADB L2T2
5816 11231 074353 SBP ER6 ROW SUBSCRIPT LEGAL?
5817 11232 025471 LDB L2T2 YES, RECALL ROW SUBSCRIPT
5818 11233 061201 JSM IMPY COMPUTE ROW DISPLACEMENT
5819 11234 025470 LDB L2T1
5820 11235 074076 TCB
5821 11236 105412 ADB LSTPT,I COLUMN SUBSCRIPT LEGAL?
5822 11237 074112 SBM *+2
5823 11240 160206 JSM ERRA,I NO, ERROR
5824 11241 101412 ADA LSTPT,I YES, ADD IN COLUMN
DISPLACEMENT
5825 11242 025413 LDB HSTPT FIND TYPE OF ARRAY
5826 11243 074070 SIB *+1
5827 11244 074537 LDB B,I
5828 11245 074042 SBR 2 FIND # OF WORDS PER ELEMENT
5829 11246 076110 RZB *+2 FULL PRECISION?
5830 11247 070704 SAL 2 YES, MULTIPLY BY 4
5831 11250 014012 CPB .1 SPLIT?
5832 11251 070744 SAL 1 YES, MULTIPLY BY 2
5833 11252 101413 ADA HSTPT,I

230

1,444,141

230

PAGE 127

FORMULA EVALUATION ROUTINES

5834	11253	131413	STA	HSTPT,I	STORE ACTUAL ADDRESS OF
					ELEMENT
5835	11254	055412	DSZ	LSTPT	UNSTACK I
5836	11255	066565	JMP	FORM1	
5837*					
5838*	CALL ADD				
5839*					
5840	11256	063376	EFAD	JSM	BINOP
5841	11257	160216	JSM	FADA,I	
5842*	PROCEED TO FOR14				
5843*					
5844*	OPERATORS CREATING INTERMEDIATE RESULTS RETURN HERE				
5845*	WITH RESULT IN AR2				
5846*					
5847	11260	020214	FOR14	LDA	AR2A
5848	11261	025411		LDB	TSTPT
5849	11262	170004		XFR	TRANSFER TO T-STACK
5850	11263	021413		LDA	HSTPT
5851	11264	070070		SIA	*+1
5852	11265	074742		SBR	16
5853	11266	070577		STB	A,I
5854	11267	025411		LDB	TSTPT
5855	11270	135413		STB	HSTPT,I
5856	11271	066637		JMP	FORM4
5857*					
5858*	CALL MULTIPLY				
5859*					
5860	11272	063376	EFMP	JSM	BINOP
5861	11273	160220		JSM	FMPA,I
5862	11274	067260		JMP	FOR14
5863*					
5864*	CALL DIVIDE				
5865*					
5866	11275	063376	EFDV	JSM	BINOP
5867	11276	160221		JSM	FDVA,I
5868	11277	067260		JMP	FOR14
5869*					
5870*	CALL EXPONENTIAL ARROW				
5871*					
5872	11300	063376	EPWR	JSM	BINOP
5873	11301	160224		JSM	UPARA,I
5874	11302	067260		JMP	FOR14
5875*					
5876*	CALL UNARY MINUS				
5877*					
5878	11303	063404	EUMIN	JSM	STTOP
5879	11304	025411		LDB	TSTPT
5880	11305	170004		XFR	TRANSFER TO T-STACK
5881	11306	000133		ADA	M2
5882	11307	070517		LDA	A,I
5883	11310	070250		SZA	*+5
5884	11311	121411		LDA	TSTPT,I
5885	11312	072111		SLA	*+2,S
5886	11313	070071		SLA	*+1,C
5887	11314	131411		STA	TSTPT,I
5888	11315	067263		JMP	FOR14+3
5889*					

PAGE 128 FORMULA EVALUATION ROUTINES

5890* EXECUTE LEFT BRACKET

5891*
 5892 11316 014002 SCCNT OCT 14002
 5893*
 5894 11317 045412 ELBRC ISZ LSTPT
 5895 11320 027316 LDB SCCNT
 5896 11321 060563 JSM SLWST
 5897 11322 063406 JSM RSCHK
 5898 11323 055413 DSZ HSTPT
 5899 11324 060553 JSM BHSTP
 5900 11325 020020 TRUE LDA ONE
 5901 11326 067261 JMP FOR14+1

STACK A
 SUBSCRIPT COMMA
 ON OPERATOR STACK
 STACK A 1 ON
 OPERATOR

STACK A 1 ON T-STACK

5902* COMPARE TOP OPERANDS OF STACK

5903*
 5904*
 5905 11327 063376 COMPR JSM BINOP
 5906 11330 160217 JSM FSBA,1
 5907 11331 020214 LDA AR2A
 5908 11332 065317 JMP SGNS

GET OPERAND ADDRESSES
 SUBTRACT ONE FROM ANOTHER
 POINT TO RESULT IN AR2
 GET SGN IN (B)

5909* EXECUTE >

5910*
 5911*
 5912 11333 063327 EGTRT JSM COMPR
 5913 11334 074753 SBP ENEQL+1
 5914 11335 067354 JMP FALSE

COMPARE OPERANDS
 <?
 YES

5915* EXECUTE <

5916*
 5917*
 5918 11336 063327 ELST JSM COMPR
 5919 11337 074056 CMB
 5920 11340 067342 JMP *+2

COMPARE OPERANDS
 REVERSE COMPARISON SENSE

5921* EXECUTE >=

5922*
 5923*
 5924 11341 063327 EGORE JSM COMPR
 5925 11342 075153 SBP TRUE
 5926 11343 067354 JMP FALSE

COMPARE OPERANDS
 <?
 YES

5927* EXECUTE =

5928*
 5929*
 5930 11344 063327 EEQL JSM COMPR
 5931 11345 075010 SZB TRUE
 5932 11346 067354 JMP FALSE

COMPARE OPERANDS
 =?
 NO

5933* EXECUTE <=

5934*
 5935*
 5936 11347 063327 ELORE JSM COMPR
 5937 11350 075653 SBP EEQL+1
 5938 11351 067325 LTRUE JMP TRUE

COMPARE OPERANDS
 >=?
 NO

5939* EXECUTE # OR <>

5940*
 5941*
 5942 11352 063327 ENEQL JSM COMPR
 5943 11353 077710 RZB LTRUE
 5944 11354 021411 FALSE LDA TSTPT
 5945 11355 170000 CLR

COMPARE OPERANDS
 #?
 CLEAR T-STACK ENTRY

PAGE 129 FORMULA EVALUATION ROUTINES

```

5946 11356 067263      JMP FOR14+3
5947*
5948* EXECUTE "OR"
5949*
5950 11357 063376 EOR   JSM BINOP
5951 11360 070070      SIA *+1
5952 11361 070517      LDA A,I      LOOK AT FIRST OPERAND
5953 11362 073350      RZA LTRUE   IF NON-ZERO, RESULT IS TRUE
5954 11363 074070 EOR2  SIB *+1      LOOK AT SECOND OPERAND
5955 11364 074537      LDB B,I
5956 11365 067353      JMP ENEQL+1
5957*
5958* EXECUTE "AND"
5959*
5960 11366 063376 EAND   JSM BINOP
5961 11367 070070      SIA *+1
5962 11370 070517      LDA A,I      LOOK AT FIRST OPERAND
5963 11371 071150      SZA FALSE   ZERO?
5964 11372 067363      JMP EOR2    NO

5966*
5967* PREPARE TO EXECUTE A BINARY OPERATOR.
5968*
5969 11373 024057 BINO1  LDB 1000
5970 11374 060406      JSM XFAR1+1  TRANSFER TO SECOND TEMP(YC)
5971 11375 067401      JMP BINO2
5972 11376 063433 BINOP  JSM OPCHK
5973 11377 014111      CPB B1740
5974 11400 067373      JMP BINO1
5975 11401 035726 BINO2  STB MBIN1
5976 11402 045413      ISZ HSTPT
5977 11403 045413      ISZ HSTPT
5978* PROCEED TO STTOP
5979*
5980* FETCH TOP OF STACK
5981*
5982 11404 063433 STTOP  JSM OPCHK
5983 11405 035471      STB L2T2      SAVE SECOND ADDRESS
5984* PROCEED TO RSCHK
5985*
5986* ALLOT SPACE FOR INTERMEDIATE RESULT
5987*
5988 11406 021411 RSCHK  LDA TSTPT      MOVE TEMPORARY POINTER
5989 11407 000021      ADA 4
5990 11410 031411      STA TSTPT
5991 11411 011410      CPA LSTAK
5992 11412 067414      JMP *+2      OVERFLOW INTO LOW STACK?
5993 11413 067430      JMP RSCH2    YES
5994 11414 031476      STA TEMP3    NO
5995 11415 000025      ADA 8
5996 11416 031410      STA LSTAK
5997 11417 021412      LDA LSTPT

```

PAGE 130

FORMULA EVALUATION ROUTINES

| | | | | | | |
|-------|-------|--------|-------|-----|---------|----------------------------------|
| 5998 | 11420 | 070070 | | SIA | *+1 | |
| 5999 | 11421 | 031475 | | STA | TEMP2 | |
| 6000 | 11422 | 000024 | | ADA | 7 | |
| 6001 | 11423 | 031412 | RSCH1 | STA | LSTPT | |
| 6002 | 11424 | 070070 | | SIA | *+1 | |
| 6003 | 11425 | 031477 | | STA | TEMP4 | |
| 6004 | 11426 | 060554 | | JSM | BHSTP+1 | TEST FOR OVERLAP WITH HIGH STACK |
| 6005 | 11427 | 061121 | | JSM | MVTOH | MOVE LOW STACK UP |
| 6006 | 11430 | 021471 | RSCH2 | LDA | L2T2 | RECALL SECOND ADDRESS |
| 6007 | 11431 | 025726 | | LDB | MBIN1 | RECALL FIRST ADDRESS |
| 6008 | 11432 | 170402 | FORMR | RET | | |
| 6009* | | | | | | |
| 6010* | | | | | | VERIFY LEGITIMACY OF OPERAND |
| 6011* | | | | | | |
| 6012 | 11433 | 125413 | OPCHK | LDB | HSTPT,1 | GET OPERAND ADDRESS |
| 6013 | 11434 | 015411 | | CPB | TSTPT | IS IT TEMPORARY? |
| 6014 | 11435 | 067460 | | JMP | OPC3 | YES |
| 6015 | 11436 | 074517 | | LDA | B,1 | NO, GET EXPONENT WORD |
| 6016 | 11437 | 010062 | | CPA | FLGBT | IS OPERAND DEFINED? |
| 6017 | 11440 | 160206 | ER8 | JSM | ERRA,1 | YES, ERROR |
| 6018 | 11441 | 021413 | | LDA | HSTPT | YES, GET TYPE |
| 6019 | 11442 | 070070 | | SIA | *+1 | |
| 6020 | 11443 | 070517 | | LDA | A,1 | |
| 6021 | 11444 | 050037 | OPCH1 | AND | 28 | GET PRECISION BITS |
| 6022 | 11445 | 071250 | | SZA | FORMR | FULL PRECISION? |
| 6023 | 11446 | 010021 | | CPA | 4 | NO, SPLIT? |
| 6024 | 11447 | 067456 | | JMP | OPC2 | YES |
| 6025 | 11450 | 070206 | | RAR | 5 | |
| 6026 | 11451 | 071052 | | SAM | FORMR | NUMBERED VARIABLE? |
| 6027 | 11452 | 074537 | | LDB | B,1 | NO, IT MUST BE INTEGER |
| 6028 | 11453 | 160227 | | JSM | FXFLA,1 | RESULT IS IN AR2 NOW |
| 6029 | 11454 | 024111 | OPC1 | LDB | BI740 | |
| 6030 | 11455 | 064406 | | JMP | XFAR1+1 | TRANSFER TO XC REGISTER |
| 6031 | 11456 | 160231 | OPC2 | JSM | SPLFA,1 | CONVERT SPLIT TO FULL PRECISION |
| 6032 | 11457 | 067454 | | JMP | OPC1 | |
| 6033 | 11460 | 021411 | OPC3 | LDA | TSTPT | DELETE TEMPORARY |
| 6034 | 11461 | 000135 | | ADA | M4 | |
| 6035 | 11462 | 031411 | | STA | TSTPT | |
| 6036 | 11463 | 170402 | | RET | | |
| 6037* | | | | | | 11464 IS LAST |
| 6038* | | | | | | MATH ROUTINES HERE |

PAGE 131

FUNCTION TABLES

| | | | | | |
|-------|-------|--------|--|------------------------|--|
| 6040 | 11707 | | ORG 11707B | | |
| 6041* | | | | | |
| 6042* | | | FUNCTION SYNTAX AND EXECUTION JUMP TABLE | | |
| 6043* | | | | | |
| 6044 | 11707 | 000002 | ABS EPI-* | | |
| 6045 | 11710 | 000003 | ABS ERES-* | | |
| 6046 | 11711 | 023715 | EPI LDA PIA | | |
| 6047 | 11712 | 064445 | JMP XFAR2+1 | | |
| 6048 | 11713 | 020215 | ERES LDA TMPOP | | |
| 6049 | 11714 | 064445 | JMP XFAR2+1 | | |
| 6050 | 11715 | 003764 | PIA DEF 3764B | | |
| 6051* | | | | | |
| 6052* | | | EXECUTE "NOT" | | |
| 6053* | | | | | |
| 6054 | 11716 | 063404 | ENOT JSM STTOP | | |
| 6055 | 11717 | 070070 | SIA *+1 | | |
| 6056 | 11720 | 070537 | LDB A,I | | |
| 6057 | 11721 | 067345 | JMP EEQL+1 | | |
| 6058* | | | | | |
| 6059 | 11722 | 000000 | BSS 13 | | |
| 6060 | 11737 | 175125 | ABS TABER-* | FUNCTION JUMP TABLE | |
| 6061* | | | | TAB | |
| 6062* | | | FUNCTION NAME TABLE | | |
| 6063* | | | | | |
| 6064 | 11740 | 042530 | FTN13 ASC 1,EX | | |
| 6065 | 11741 | 050202 | OCT 050202 | EXP; OP CODE 2 | |
| 6066 | 11742 | 046117 | ASC 1,LO | | |
| 6067 | 11743 | 043603 | OCT 43603 | LOG; OP CODE 3 | |
| 6068 | 11744 | 040502 | ASC 1,AB | | |
| 6069 | 11745 | 051604 | OCT 051604 | ABS; OP CODE 4 | |
| 6070 | 11746 | 051521 | ASC 1,SQ | | |
| 6071 | 11747 | 051205 | OCT 051205 | SQR; OP CODE 5 | |
| 6072 | 11750 | 044516 | ASC 1,IN | | |
| 6073 | 11751 | 052206 | OCT 052206 | INT; OP CODE 6 | |
| 6074 | 11752 | 051507 | ASC 1,SG | | |
| 6075 | 11753 | 047207 | OCT 47207 | SGN; 7 | |
| 6076 | 11754 | 051511 | ASC 1,SI | | |
| 6077 | 11755 | 047210 | OCT 047210 | SIN; 10 | |
| 6078 | 11756 | 041517 | ASC 1,CO | | |
| 6079 | 11757 | 051611 | OCT 051611 | COS; 11 | |
| 6080 | 11760 | 052101 | ASC 1,TA | | |
| 6081 | 11761 | 047212 | OCT 047212 | TAN; 12 | |
| 6082 | 11762 | 040524 | ASC 1,AT | | |
| 6083 | 11763 | 047213 | OCT 047213 | ATN; 13 | |
| 6084 | 11764 | 046107 | ASC 1,LG | | |
| 6085 | 11765 | 052214 | OCT 52214 | LGT; 14 | |
| 6086 | 11766 | 040523 | ASC 1,AS | | |
| 6087 | 11767 | 047215 | OCT 47215 | ASN; 15 | |
| 6088 | 11770 | 040503 | ASC 1,AC | | |
| 6089 | 11771 | 051616 | OCT 51616 | ACS; 16 | |
| 6090 | 11772 | 051105 | ASC 1,RE | | |
| 6091 | 11773 | 051630 | OCT 51630 | RES; 30 (NO ARGUEMENT) | |
| 6092 | 11774 | 050111 | ASC 1,PI | | |
| 6093 | 11775 | 154524 | OCT 154524 | PI; 31 (NO ARGUEMENT) | |
| 6094 | 11776 | 040502 | OCT 40502 | | |
| 6095 | 11777 | 140777 | OCT 140777 | TAB; 1 | |

247

PAGE 132

FORMULA OPERATOR TABLE

6097 12000 ORG 12000B
6098 12000 006774 STEPA DEF FOPST+FOPST
6099*

6100* FORMULA OPERATOR SEARCH AND PRECEDENCE TABLE
6101* LEFT CHAR IS ASCII; RIGHT CHAR IS PRECEDENCE
6102*

| | | | | | | OP CODE 1 |
|-------|-------|--------|-------|-----|-------|---------------------------------|
| 6103 | 12001 | 021000 | QUOT | OCT | 21000 | 2 |
| 6104 | 12002 | 026000 | COMMA | OCT | 26000 | 3 |
| 6105 | 12003 | 035400 | SMCLN | OCT | 35400 | 4 |
| 6106 | 12004 | 024400 | RPARN | OCT | 24400 | 5 |
| 6107 | 12005 | 056400 | RBRAC | OCT | 56400 | 6 |
| 6108 | 12006 | 026002 | SCMMA | OCT | 26002 | 7 |
| 6109 | 12007 | 036402 | ASSOP | OCT | 36402 | |
| 6110 | 12010 | 177377 | M257 | DEC | -257 | THIS IS NOT USED AS TABLE ENTRY |
| 6111 | 12011 | 025407 | PLUSS | OCT | 25407 | 11 |
| 6112 | 12012 | 026407 | MINS | OCT | 26407 | 12 |
| 6113 | 12013 | 025010 | TIMES | OCT | 25010 | 13 |
| 6114 | 12014 | 027410 | DIV | OCT | 27410 | 14 |
| 6115 | 12015 | 057013 | EXPS | OCT | 57013 | 15 |
| 6116 | 12016 | 037005 | GTR | OCT | 37005 | 16 |
| 6117 | 12017 | 036005 | LSS | OCT | 36005 | 17 |
| 6118 | 12020 | 021405 | UNEQL | OCT | 21405 | 20 |
| 6119 | 12021 | 036405 | EQUAL | OCT | 36405 | 21 |
| 6120 | 12022 | 026412 | UNMIN | OCT | 26412 | 22 |
| 6121 | 12023 | 025412 | UPLUS | OCT | 25412 | 23 |
| 6122 | 12024 | 055420 | LBRAC | OCT | 55420 | 24 |
| 6123 | 12025 | 024020 | LPARN | OCT | 24020 | 25 |
| 6124 | 12026 | 000003 | OROP | OCT | 00003 | 26 |
| 6125 | 12027 | 000004 | ANDOP | OCT | 00004 | 27 |
| 6126 | 12030 | 000012 | NOTOP | OCT | 00012 | 30 |
| 6127 | 12031 | 000005 | GTREQ | OCT | 00005 | 31 |
| 6128 | 12032 | 000005 | LSSEQ | OCT | 00005 | 32 |
| 6129* | | | | | | |
| 6130 | 12033 | 000000 | BSS | 1 | | *****CHECKSUM***** |

| | | | | | |
|-------|---|--------|------|-----|---|
| 6132 | | LST | | | |
| 6133 | 12034 | 007001 | TOA | DEF | FOPST,+FOPST+5 |
| 6134 | 12035 | 007004 | ORA | DEF | OR+OR |
| 6135 | 12036 | 007013 | NOTA | DEF | OR+OR+7 |
| 6136 | 12037 | 007017 | GTEA | DEF | GTE+GTE+1 |
| 6137* | | | | | |
| 6138* | CHECK SYNTAX OF A FORMULA | | | | |
| 6139* | | | | | |
| 6140* | EXIT TO ERROR ON UNRECOGNISABLE OR INCORRECT INPUT | | | | |
| 6141* | ON EXIT (A)= NEXT CHARACTER NOT IN FORMULA | | | | |
| 6142* | | | | | (B)=0 |
| 6143* | IF FIRST OPERAND IS A STRING VARIABLE, EXIT TO ERROR | | | | |
| 6144* | UNLESS STRING IS ENABLED BY SFLAG=SBPTR, THEN EXIT (B)=SFLAG=-1 | | | | |
| 6145* | | | | | |
| 6146* | MSFLG=8; LOOKING FOR "=" IN LET OR FOR STATEMENT | | | | |
| 6147* | MSFLG=9; INTERPRET "=" AS A LOGICAL OPERATOR | | | | |
| 6148* | IF A STORE IS FOUND, SFLAG=-2 | | | | |
| 6149* | | | | | |
| 6150 | 12040 | 020012 | FSC | LDA | .1 SET LEFT PARENTHESIS |
| 6151 | 12041 | 131472 | | STA | TEMPS,1 COUNT TO 0 (OFFSET BY 1) |
| 6152 | 12042 | 020132 | FSC1 | LDA | M1 ENABLE UNARY |
| 6153 | 12043 | 031713 | | STA | UFLAG OPERATORS |
| 6154* | LOOK FOR AN OPERAND | | | | |
| 6155 | 12044 | 062231 | FSC2 | JSM | VAROP LOOK FOR VARIABLE OPERAND |
| 6156 | 12045 | 066114 | | JMP | FSC7 FIRST CHAR NOT A LETTER |
| 6157 | 12046 | 066164 | | JMP | FSC13-3 SUBSCRIPTED OR STRING VARIABLE |
| 6158 | 12047 | 060752 | | JSM | LETCK SIMPLE VARIABLE; FOLLOWED BY |
| 6159 | 12050 | 066167 | | JMP | FSC13 NO LETTER? |
| 6160* | DOES "AND" OR "OR" FOLLOW SIMPLE VARIABLE? | | | | |
| 6161 | 12051 | 062217 | | JSM | MCBCK CHECK FOR "AND" OR "OR" ONLY: (B) |
| 6162* | CHECK FOR FUNCTION INSTEAD OF SIMPLE VARIABLE | | | | |
| 6163 | 12052 | 062424 | | JSM | FNCK "FN"? |
| 6164 | 12053 | 066061 | | JMP | FSC4 NO |
| 6165 | 12054 | 055400 | | DSZ | BADDR YES |
| 6166 | 12055 | 000154 | | ADA | M64 GET FUNCTION NAME |
| 6167 | 12056 | 074742 | | SBR | 16 |
| 6168 | 12057 | 135706 | | STB | SBPTR,1 CLEAR OUT NEXT WORD |
| 6169 | 12060 | 066074 | | JMP | FSC3-1 |
| 6170 | 12061 | 070342 | FSC4 | SAR | 8 RECALL LAST CHAR |
| 6171 | 12062 | 055400 | | DSZ | BADDR BACK UP TO POINT AT SECOND CHAR |
| 6172 | 12063 | 024133 | | LDB | M2 SEARCH ALL TABLES FOR |
| 6173 | 12064 | 160237 | | JSM | SALTA,1 A PREDEFINED FUNCTION |
| 6174 | 12065 | 066143 | | JMP | FSC11 NOT FOUND |
| 6175 | 12066 | 135706 | | STB | SBPTR,1 STORE TABLE CODE |
| 6176 | 12067 | 070137 | | LDB | A |
| 6177 | 12070 | 004147 | | ADB | M23 |
| 6178 | 12071 | 074076 | | TCB | |
| 6179 | 12072 | 035474 | | STB | TEMPI SET NEGATIVE IF NO ARGUMENT |
| 6180 | 12073 | 040116 | | IOR | QTOP SET FLAG FOR PRE-DEF FUNCTION |
| 6181 | 12074 | 070544 | | SAL | 5 POSITION OP CODE |
| 6182* | ASSEMBLE AND STORE FUNCTION | | | | |
| 6183 | 12075 | 040040 | FSC3 | IOR | B37 SET OPERAND TYPE FOR FUNCTION |
| 6184 | 12076 | 070137 | | LDB | A |
| 6185 | 12077 | 055706 | | DSZ | SBPTR REPLACE |
| 6186 | 12100 | 060641 | | JSM | RESBP+1 PREVIOUS OPERAND |
| 6187 | 12101 | 074217 | | IOR | B WITH FUNCTION |

PAGE 134

SYNTAX SUBROUTINES

| | | | | | |
|-------|---|--------|-------|---------|------------------------------|
| 6188 | 12102 | 131706 | STA | SBPTR,I | |
| 6189 | 12103 | 045706 | ISZ | SBPTR | |
| 6190 | 12104 | 021474 | LDA | TEMP1 | RECALL ARGUEMENT FLAG |
| 6191 | 12105 | 070253 | SAP | FSC31 | NORMAL ARITHMETIC ARGUEMENT? |
| 6192 | 12106 | 161473 | JSM | TEMP,I | NO, JUMP THROUGH SYNTAX/EXEC |
| | | | | | TABLE |
| 6193 | 12107 | 045706 | ISZ | SBPTR | RETURN HERE IF NO ARGUEMENT |
| 6194 | 12110 | 061055 | JSM | GNEXT | |
| 6195 | 12111 | 066130 | JMP | FSC8 | NO, CHECK FOR EOL NEXT |
| 6196 | 12112 | 031713 | FSC31 | STA | UFLAG |
| 6197 | 12113 | 066140 | JMP | FSC10 | DISABLE UNARY OPERATORS |
| 6198* | CHECK FOR LEFT PARENTHESIS | | | | |
| 6199 | 12114 | 045706 | FSC7 | ISZ | SBPTR |
| 6200 | 12115 | 060513 | JSM | LPCK | CHECK FOR PAREN |
| 6201 | 12116 | 066123 | JMP | *+5 | NOT FOUND |
| 6202 | 12117 | 145472 | FSC5 | ISZ | TEMPS,I |
| 6203 | 12120 | 024026 | FSC6 | LDB | .9 |
| 6204 | 12121 | 035710 | STB | MSFLG | DISABLE ASSIGNMENT |
| 6205 | 12122 | 066042 | JMP | FSC1 | |
| 6206* | CHECK FOR A NUMBER | | | | |
| 6207 | 12123 | 074742 | SBR | 16 | INITIALIZE |
| 6208 | 12124 | 035721 | STB | SIGN | TO POSITIVE |
| 6209 | 12125 | 160243 | JSM | NUMCA,I | FETCH AND BUILD CONSTANT |
| 6210 | 12126 | 066133 | JMP | FSC9 | NOT FOUND |
| 6211 | 12127 | 160244 | JSM | NUMOA,I | STORE IN SYNTAX BUFFER |
| 6212 | 12130 | 024026 | FSC8 | LDB | .9 |
| 6213 | 12131 | 035710 | STB | MSFLG | DISABLE ASSIGNMENT |
| 6214 | 12132 | 066167 | JMP | FSC13 | |
| 6215* | CHECK FOR A UNARY OPERATOR | | | | |
| 6216 | 12133 | 045713 | FSC9 | ISZ | UFLAG |
| 6217 | 12134 | 160206 | FSCE3 | JSM | ERRA,I |
| 6218 | 12135 | 060461 | JSM | SYMC2 | UNARY MINUS |
| 6219 | 12136 | 012021 | DEF | UNMIN-1 | OR UNARY PLUS? |
| 6220 | 12137 | 066134 | JMP | FSCE3 | NO |
| 6221 | 12140 | 024026 | FSC10 | LDB | .9 |
| 6222 | 12141 | 035710 | STB | MSFLG | DISABLE ASSIGNMENT |
| 6223 | 12142 | 066044 | JMP | FSC2 | |
| 6224* | CHECK FOR "NOT" RATHER THAN SIMPLE VARIABLE | | | | |
| 6225 | 12143 | 045713 | FSC11 | ISZ | UFLAG |
| 6226 | 12144 | 066212 | JMP | FSC14-1 | "NOT" PERMITTED? |
| 6227 | 12145 | 026036 | LDB | NOTA | NO |
| 6228 | 12146 | 060537 | JSM | STSRH | SEARCH FOR "NOT" |
| 6229 | 12147 | 066212 | JMP | FSC14-1 | NOT FOUND |
| 6230 | 12150 | 055706 | DSZ | SBPTR | ERASE |
| 6231 | 12151 | 060641 | JSM | RESBP+1 | SPURIOUS |
| 6232 | 12152 | 045706 | ISZ | SBPTR | OPERAND |
| 6233 | 12153 | 066140 | JMP | FSC10 | |
| 6234* | | | | | |
| 6235* | CHECK FOR RIGHT PARENTHESIS | | | | |
| 6236* | | | | | |
| 6237 | 12154 | 060524 | FSC12 | JSM | RPCK |
| 6238 | 12155 | 066213 | JMP | FSC14 | RIGHT PAREN? |
| 6239 | 12156 | 155472 | DSZ | TEMPS,I | NO |
| 6240 | 12157 | 066167 | JMP | FSC13 | ((A)=NEXT CHAR) |
| 6241 | 12160 | 055706 | DSZ | SBPTR | NO MATCHING LEFT PAREN |
| 6242 | 12161 | 055400 | DSZ | BADDR | RESTORE |
| 6243 | 12162 | 020046 | LDA | .41 | PREVIOUS CONDITIONS |

PAGE 135

SYNTAX SUBROUTINES

```

6244 12163 066215      JMP FSC14+2      AND RETURN
6245* CHECK FOR A BINARY OPERATOR
6246 12164 024132      LDB M1      STRING VARIABLE
6247 12165 015714      CPB SFLAG    JUST FOUND?
6248 12166 170402      RET          YES
6249 12167 010031      FSC13      CPA EOL      END OF STMT?
6250 12170 066213      JMP FSC14      YES
6251 12171 031713      STA UFLAG    DISABLE UNARY OPERATORS
6252 12172 031474      STA TEMP1    SAVE CHAR
6253 12173 024132      LDB M1
6254 12174 062217      JSM MCBCK    CHECK FOR AND, OR, >=, <=, <>
6255 12175 025710      LDB MSFLG
6256 12176 060464      JSM SYMCK    SEARCH FOR
6257 12177 021010      DEF PLUS-1    SINGLE CHAR OPERATOR
6258 12200 066204      JMP *+4      NOT FOUND
6259 12201 010124      CPA EXPOP    EXPONENTIATION?
6260 12202 066140      JMP FSC10      YES
6261 12203 066120      JMP FSC6      NO
6262 12204 060463      JSM SYMC1    NO, "=" ASSIGNMENT?
6263 12205 012006      DEF ASSOP-1
6264 12206 066154      JMP FSC12      NO
6265 12207 020133      LDA M2      YES, SET
6266 12210 031714      STA SFLAG    STORE OCCURRED FLAG
6267 12211 066042      JMP FSC1
6268* END OF FORMULA
6269 12212 061055      JSM GNEXT    RECALL CHAR
6270 12213 155472      FSC14      DSZ TEMPS,I  ALL PARENS MATCHED?
6271 12214 064535      JMP FSCE2    MISSING RIGHT PAREN
6272 12215 074742      SBR 16
6273 12216 170402      RET          RETURN WITH (B)=0
6274*
6275* CHECK FOR A MULTI-CHARACTER BINARY OPERATOR
6276*
6277* (B)=1; CHECK >=, <=, <> AS WELL AS AND, OR
6278* EXIT TO FSC6 IF FOUND
6279*
6280 12217 135706      MCBCK      STB SBPTR,I  SAVE FLAG
6281 12220 026035      LDB ORA      SEARCH FOR "AND" OR "OR"
6282 12221 060537      JSM STSRH
6283 12222 066225      JMP *+3      NOT FOUND
6284 12223 055777      DSZ RSPTR    DINC STACK POINTER
6285 12224 066120      JMP FSC6      RETURN
6286 12225 145706      ISZ SBPTR,I  CHECK REST OR OPERATORS
6287 12226 170402      RET          NO
6288 12227 026037      LDB GTEA    CHECK FOR >=, <=, <>
6289 12230 066221      JMP MCBCK+2

```

PAGE 136

SYNTAX SUBROUTINES

6291*
 6292* SEEK A VARIABLE OPERAND
 6293* P+1: FIRST CHAR NOT A LETTER
 6294* P+2: ARRAY OR STRING VARIABLE FOUND
 6295* P+3: SIMPLE VARIABLE
 6296*
 6297* (A)=NEXT CHAR FOLLOWING OPERAND
 6298* IF STRING IS FOUND AND SFLAG=SBPTR THEN SET SFLAG=-1
 6299* IF SFLAG#SBPTR THEN STRING IS NOT PERMITTED; ERROR
 6300*
 6301 12231 061140 VAROP JSM LTR LETTER?
 6302 12232 170402 RET NO, RETURN TO P+1
 6303 12233 074742 SBR 16 INITIALIZE FOR
 6304 12234 035717 STB OPTYP FULL PRECISION
 6305 12235 010077 CPA B44 IS NEXT CHAR "S"?
 6306 12236 066275 JMP VARO4 YES
 6307 12237 060750 JSM DIGCK DIGIT?
 6308 12240 066250 JMP VARO1 NO
 6309 12241 070137 LDB A YES, RESTORE ASCII
 6310 12242 021474 LDA TEMP1 RECALL LETTER
 6311 12243 035474 STB TEMP1 SAVE DIGIT
 6312 12244 060450 JSM STROP COMBINE AND STORE OPERAND
 6313 12245 061055 JSM GNEXT
 6314 12246 031475 STA TEMP2 SAVE IT
 6315 12247 064356 JMP RETN3
 6316 12250 074742 VARO1 SBR 16
 6317 12251 010073 CPA S
 6318 12252 224021 LDB .4 OPTYP=4 FOR SPLIT PRECISION
 6319 12253 010055 CPA 1
 6320 12254 024025 LDB .8 OPTYP=8 FOR INTEGER
 6321 12255 074350 SZB VARO6 SKIP IF FULL PRECISION
 6322 12256 055771 DSZ DFLAG DIM OR COM?
 6323 12257 066261 JMP *+2 NO, SPLIT OR INTEGER INT ALLOWED
 6324 12260 035717 STB OPTYP YES, STORE OPERAND TYPE
 6325 12261 045711 ISZ DFLAG DIM OR COM?
 6326 12262 061055 JSM GNEXT YES GET CHAR FOLLOWING PRECISION
 6327 12263 031475 STA TEMP2 STORE CHAR
 6328 12264 010045 VARO6 CPA .40 "("
 6329 12265 066275 JMP VARO4 YES
 6330 12266 010102 CPA B133 "["?
 6331 12267 066275 JMP VARO4 YES
 6332 12270 021474 LDA TEMP1 RECALL LETTER
 6333 12271 025717 LDB OPTYP
 6334 12272 060451 JSM STROP+1 COMBINE AND STORE SIMPLE VARIABLE
 6335 12273 021475 VARO3 LDA TEMP2 NO, RECALL NEXT CHAR
 6336 12274 064356 JMP RETN3 RETURN TO P+3
 6337 12275 062436 VARO4 JSM STRID STORE STRING OR ARRAY OPERAND
 6338 12276 062301 JSM SBSCK GET SUBSCRIPT
 6339 12277 074742 SBR 16 NO SUBSCRIPT, SET B=0
 6340 12300 064357 JMP RETN2

PAGE 137

SYNTAX SUBROUTINES

6342*
6343* SUBSCRIPT SYNTAX
6344*
6345* ARYAD POINTS TO ARRAY OPERAND
6346* EXIT TO P+1 IF FIRST CHAR ISNOT "("OR "[" OTHERWISE P+2
6347* DFLAG=1 IF ANALYSING DIM OR COM STMT; 0 OTHERWISE
6348* SFLAG=1 FOR STRING OPERAND
6349*
6350* ARRAY OPERANDS ARE MARKED AS ONE OR TWO DIMENITIONAL
6351* MEMORY REQUIREMENT IS COMPUTED FOR COM STMT
6352*
6353 12301 060513 SBSCK JSM LPCK FIRST CHAR A "("OR "["?
6354 12302 170402 RET NO
6355 12303 020125 LDA LBOP YES, RECORD
6356 12304 131706 STA SBPTR,I A LEFT BRACKET
6357 12305 155723 DSZ ARYAD, SET ARRAY
6358 12306 155723 DSZ ARYAD,I TO / DIMENSION
6359 12307 021711 LDA DFLAG DIM OR COM?
6360 12310 072110 RZA *+2
6361 12311 066363 JMP SBSC3
6362 12312 024157 LDB M256 ALLOW 255
6363 12313 045714 ISZ SFLAG STRING?
6364 12314 004132 ADB M1 NO, ALLOW UP TO 256
6365 12315 060666 JSM PGINT GET AN INTEGER
6366 12316 004132 ADB M1 OFFSET BY -1
6367 12317 074404 SBL 8
6368 12320 035474 STB TEMP1 SAVE DIMENSION
6369 12321 025714 LDB SFLAG
6370 12322 074450 SZB SBSC1 SKIP IF STRING
6371 12323 060463 JSM SYMC1 LOOK FOR A COMMA
6372 12324 012005 DEF SCMMA-1 A COMMA
6373 12325 066333 JMP SBSC1 NOT FOUND
6374 12326 145732 ISZ ARYAD, I SET TO 2 DIMENSIONS
6375 12327 026010 LDB M257 GET SECOND
6376 12330 060666 JSM PGINT DIMENSION
6377 12331 004132 ADM M1 OFFSET BY -1
6378 12332 066334 JMP SBSC1+1
6379 12333 074742 SBR 16 SECOND DIM=1 OFFSET BY -1
6380 12334 005474 ADB TEMP1 COMBINE DIMENSIONS
6381 12335 035714 STB SFLAG SAVE PACKED DIMENSIONS
6382 12336 060534 JSM RPCKE DEMAND ")" OR "["
6383 12337 024123 LDB RBOP STORE
6384 12340 055706 DSZ SBPTR
6385 12341 135706 STB SBPTR,I A RIGIT BRACKET
6386 12342 045706 ISZ SBPTR
6387 12343 025711 LDB DFLAG
6388 12344 076710 RZB SBSCR DIM OR COM?
6389* RESTORE LOCAL QUANTITIES FOR FSC
6390 12345 045472 ISZ TEMPS
6391 12346 024112 LDB RSTKA RESET
6392 12347 035777 STB RSPTR STACK POINTER
6393 12350 066353 JMP FPOPI
6394 12351 135777 STB RSPTR,I STORE ON RETURN STACK
6395 12352 045777 ISZ RSPTR
6396 12353 124572 LDB TEMPS,I GET WORD FROM S—STACK
6397 12354 045472 ISZ TEMPS

PAGE 138

SYNTAX SUBROUTINES

| | | | | | |
|-------|--|--------|-------|-----------|-------------------------------|
| 6398 | 12355 | 075633 | SBP | FPOP2,C | IF NEGATIVE: IT MUST BE MSFLG |
| 6399 | 12356 | 035710 | STB | MSFLG | RESTORE MSFLG |
| 6400 | 12357 | 125472 | LDB | TEMPS,I | |
| 6401 | 12360 | 045472 | ISZ | TEMPS | |
| 6402 | 12361 | 035713 | STB | UFLAG | RESTORE UFLAG |
| 6403 | 12362 | 064357 | SBSCR | JMP RETN2 | RETURN TO P+2 |
| 6404* | SAVE LOCAL QUANTITIES FOR FSC | | | | |
| 6405 | 12363 | 021713 | SBSC3 | LDA UFLAG | |
| 6406 | 12364 | 062416 | JSM | DINCT | SAVE UFLAG ON THE S-STACK |
| 6407 | 12365 | 021710 | LDA | MSFLG | |
| 6408 | 12366 | 072053 | SAP | *+1,S | SET FLAG |
| 6409 | 12367 | 062416 | JSM | DINCT | SAVE MSFLG |
| 6410 | 12370 | 055777 | FRCU1 | DSZ RSPTR | |
| 6411 | 12371 | 121777 | LDA | RSPTR,I | GET RETURN ADDRESS |
| 6412 | 12372 | 062416 | JSM | DINCT | SAVE IT |
| 6413 | 12373 | 021777 | LDA | RSPTR | |
| 6414 | 12374 | 010112 | CPA | RSTKA | ALL OF STACK MOVED? |
| 6415 | 12375 | 066377 | JMP | *+2 | YES |
| 6416 | 12376 | 066370 | JMP | FRCU1 | |
| 6417 | 12377 | 021723 | LDA | ARYAD | |
| 6418 | 12400 | 062416 | JSM | DINCT | SAVE ARRAY ADDRESS |
| 6419 | 12401 | 062416 | JSM | DINCT | DINK AGAIN AND CHECK OVERFLOW |
| 6420 | 12402 | 020026 | LDA | .9 | DISABLE STORE MODE |
| 6421 | 12403 | 031710 | STA | MSFLG | |
| 6422 | 12404 | 062040 | JSM | FSC | GET INDEX FORMULA |
| 6423 | 12405 | 045472 | ISZ | TEMPS | RESTORE |
| 6424 | 12406 | 125472 | LDB | TEMPS,I | ARRAY |
| 6425 | 12407 | 035723 | STB | ARYAD | ADDRESS |
| 6426 | 12410 | 060463 | JSM | SYMCI | |
| 6427 | 12411 | 012005 | DEF | SCMMA-1 | COMMA? |
| 6428 | 12412 | 066336 | JMP | SBSC2 | NO |
| 6429 | 12413 | 145723 | ISZ | ARYAD,I | SET TO 2 DIMENSIONS |
| 6430 | 12414 | 062040 | JSM | FSC | GET SECOND INDEX FORMULA |
| 6431 | 12415 | 066336 | JMP | SBSC2 | |
| 6432* | | | | | |
| 6433* | DECREMENT TEMPS AND CHECK FOR OVERFLOW | | | | |
| 6434* | | | | | |
| 6435 | 12416 | 055472 | DINCT | DSZ TEMPS | |
| 6436 | 12417 | 025472 | LDB | TEMPS | |
| 6437 | 12420 | 015412 | CPB | LSTPT | OVERFLOW? |
| 6438 | 12421 | 064561 | JMP | FSCE4 | YES |
| 6439 | 12422 | 131472 | STA | TEMPS,I | NO, SAVE (A) ON STACK |
| 6440 | 12423 | 170402 | RET | | |

PAGE 139

SYNTAX SUBROUTINES

```

6442*
6443* CHECK FOR "FN" FOLLOWED BY A LETTER
6444* EXIT P+1; NOT FOUND
6445* P+2; FOUND, (A)=LETTER
6446*
6447 12424 025474 FNCK LDB TEMP1
6448 12425 074404 SBL 8
6449 12426 074217 IOR B
6450 12427 010162 CPA FN "FN"?
6451 12430 066432 JMP *+2 YES
6452 12431 170402 RET
6453 12432 061140 JSM LTR NEXT CHAR A LETTER?
6454 12433 160206 SYNE4 JSM ERRA,I NO
6455 12434 021474 LDA TEMP1 YES, RECALL LETTER
6456 12435 064357 JMP RETN2
6457*
6458* RECORD A STRING OR ARRAY IDENTIFIER
6459*
6460 12436 025706 STRID LDB SBPTR SET POINTER TO
6461 12437 035723 STB ARYAD ARRAY OR STRING OPERAND
6462 12440 010077 CPA B44 NEXT CHAR "S"?
6463 12441 066450 JMP STRI1 YES
6464 12442 021474 LDA TEMP1 GET OPERAND LETTER
6465 12443 025717 LDB OPTYP SET THE TYPE TO FULL, SPLIT, OR
INTEGER
6466 12444 004020 ADB 3 WITH UNKNOWN DIMENSIONS
6467 12445 060451 JSM STROP+1 COMBINE AND STORE
6468 12446 021475 LDA TEMP2 RECALL FOLLOWING CHAR
6469 12447 170402 RET
6470 12450 015714 STRI1 CPB SFLAG STRING VARIABLE ALLOWED?
6471 12451 066453 JMP *+2 YES
6472 12452 160206 STRNP JSM ERRA,I NO
6473 12453 020132 LDA M1 SET FLAG
6474 12454 031714 STA FLAG TO "STRING OCCURRED"
6475 12455 021474 LDA TEMP1
6476 12456 024076 LDB B36
6477 12457 035723 STB ARYAD POINT ADDRESS AT ROM
6478 12460 060451 JSM STROP+1 STORE STRING OPERAND
6479 12461 065055 JMP GNEXT GET NEXT CHAR AND RETURN

```

255

PAGE 140 SYNTAX

6481*
6482* FOR STATEMENT SYNTAX
6483*
6484 12462 062231 FORS JSM VAROP SEEK A SIMPLE VARIABLE
6485 12463 066464 JMP SYNE8
6486 12464 160206 SYNEB JSM ERRA,I NOT FOUND
6487 12465 060463 JSM SYMC1 SEEK "="
6488 12466 012006 DEF ASSOP-1
6489 12467 066522 JMP SYNE6 NOT FOUND
6490 12470 062040 JSM FSC GET REST OF INITIAL FORMULA
6491 12471 026034 LDB TOA SEEK "TO"
6492 12472 060537 JSM STSRH
6493 12473 160206 SYNE9 JSM ERRA,I NOT FOUND
6494 12474 062040 JSM FSC GET ENDING FORMULA
6495 12475 061153 JSM EOLCK EOL?
6496 12476 026000 FORSI LDB STEPA SEEK "STEP"
6497 12477 060537 JSM STSRH
6498 12500 160236 SYE10 JSM ERRA,I NOT FOUND
6499*
6500* WAIT STATEMENT SYNTAX
6501*
6502 12501 062040 WAITS JSM FSC GET FORMULA
6503 12502 065151 JMP EOST DEMAND END OF STMT
6504*
6505* DEF STATEMENT SYNTAX
6506*
6507 12503 061140 DEFS JSM LTR
6508 12504 066433 JMP SYNE4
6509 12505 062424 JSM FNCK "FN" FOLLOWED BY LETTER?
6510 12506 066433 JMP SYNE4 NO
6511 12507 024040 LDB B37 YES RECORD
6512 12510 060451 JSM STROP+1 FUNCTION NAME
6513 12511 021475 LDA TEMP2
6514 12512 060521 JSM LPCKE DEMAND LEFT PAREN
6515 12513 062231 JSM VAROP FETCH SIMPLE VARIABLE
6516 12514 066515 JMP SYNE5
6517 12515 160206 SYNES JSM ERRA,I NOT FOUND
6518 12516 060534 JSM RPCKE DEMAND RIGHT PAREN
6519 12517 061153 JSM EOLCK EOL?
6520 12520 060463 JSM SYMC1 DEMAND
6521 12521 012006 DEF ASSOP-1 AN "="
6522 12522 160206 SYNE6 JSM ERRA,I
6523 12523 066501 JMP WAITS FETCH ASSIGNING FORMULA
6524*
6525* RETURN STATEMENT SYNTAX
6526*
6527 12524 061066 RETNS JSM GETSK RETURN A VALUE?
6528 12525 065150 JMP EOST-1 NO
6529 12526 055400 DSZ BADDR
6530 12527 066501 JMP WAITS YES, GET A FORMULA
6531* 12533 IS LAST
6532* MATH ROUTINES HERE

256

PAGE 141

STATEMENT EXECUTION JUMP TABLE

6534*

6535* EXECUTION JUMP TABLE

6536*

| | | | | | |
|------|-------|--------|------|---------|-------------|
| 6537 | 13741 | | ORG | 13641B | |
| 6538 | 13741 | 173131 | ABS | EWRT-* | |
| 6539 | 13742 | 173772 | ABS | EGRAD-* | |
| 6540 | 13743 | 173767 | ABS | EDEG-* | |
| 6541 | 13744 | 173763 | ABS | ERAD-* | |
| 6542 | 13745 | 174111 | ABS | ELET-* | |
| 6543 | 13746 | 174044 | ABS | XEC4-* | FORMAT |
| 6544 | 13747 | 172777 | DISP | EDISP-* | |
| 6545 | 13750 | 174505 | ABS | EINPT-* | |
| 6546 | 13751 | 174562 | ABS | ERSTR-* | |
| 6547 | 13752 | 172773 | ABS | EPRIN-* | |
| 6548 | 13753 | 174011 | ABS | EREAD-* | |
| 6549 | 13754 | 174036 | ABS | XEC4-* | DATA |
| 6550 | 13755 | 173531 | ABS | EWAIT-* | |
| 6551 | 13756 | 174130 | ABS | ESTOP-* | |
| 6552 | 13757 | 174106 | ABS | EEND-* | |
| 6553 | 13760 | 174252 | ABS | ERTRN-* | |
| 6554 | 13761 | 002577 | ABS | EGOSB-* | |
| 6555 | 13762 | 174354 | ABS | ENEXT-* | |
| 6556 | 13763 | 174265 | ABS | EFOR-* | |
| 6557 | 13764 | 174213 | ABS | EIF-* | |
| 6558 | 13765 | 002574 | ABS | EGOTO-* | |
| 6559 | 13766 | 174024 | ABS | XEC4-* | REM |
| 6560 | 13767 | 174023 | ABS | XEC4-* | DEF |
| 6561 | 13770 | 174022 | ABS | XEC4-* | COM |
| 6562 | 13771 | 174021 | ABS | XEC4-* | DIM |
| 6563 | 13772 | 174064 | ABS | ELET-* | IMPLIED LET |

| | | | | | |
|------|-------|--------|-----|---------|----------------------------|
| 6565 | 13773 | 167371 | ABS | FOP13-* | NON-FORMULA OPERATOR TABLE |
| 6566 | 13774 | 175744 | ABS | FTN13-* | FUNCTION TABLE |
| 6567 | 13775 | 167434 | ABS | CMD13-* | SYSTEM COMMAND TABLE |
| 6568 | 13776 | 173560 | ABS | SYN13-* | STATEMENT TABLE |
| 6569 | 13777 | 012000 | OCT | 12000 | TABLE OP-CODE |

PAGE 142 SYNTAX

6571 16410 ORG 16410B
6572*
6573* CHECK SYNTAX OF DIM AND COM STATEMENTS
6574*
6575 16410 045711 ARRYS ISZ DFLAG SET DIM MODE FOR SBSCK
6576 16411 021706 LDA SBPTR
6577 16412 031710 STA MSFLG SAVE POINTER
6578 16413 031714 STA SFLAG ENABLE STRING VARIABLES
6579 16414 160246 JSM VAROA,I FETCH I.D.
6580 16415 164321 JMP E25+1,I MISSING VARIABLE
6581 16416 074610 SZB ARRE1 ARRAY OR STRING DIMENSIONED?
6582 16417 031475 STA TEMP2
6583 16420 121710 LDA MSFLG,I FETCH IDENTIFIER
6584 16421 025714 LDB SFLAG FETCH PACKED DIMENSIONS
6585 16422 160240 JSM MDIMA,I COMPUTE SPACE REQUIRED
6586 16423 001461 ADA FTMP2 UPDATE
6587 16424 031461 STA FTMP2 COMMON POINTER
6588 16425 021475 LDA TEMP2
6589 16426 010031 ARRY1 CPA EOL END OF STMT?
6590 16427 170402 RET YES
6591 16430 060510 JSM COMCE DEMAND A COMMA
6592 16431 066411 JMP ARRY5+1 CONTINUE
6593 16432 160206 ARRE1 JSM ERRA,I NO DIMENSION ON STRING
6594*
6595* COM STMT SYNTAX AND STORAGE ALLOCATION
6596*
6597 16433 025423 COMS LDB PBPTR
6598 16434 015424 CPB PBUFF NULL PROGRAM?
6599 16435 066437 JMP *+2
6600 16436 160206 SYNE3 JSM ERRA,I NO, ILLEGAL COM
6601 16437 021422 LDA UKPTR
6602 16440 072710 RZA DIMS SKIP IF KEY; NO STORAGE ALLOCATION

6603 16441 035461 STB FTMP2 INITIALIZE COMMON POINTER
6604 16442 062410 JSM ARRYS CHECK SYNTAX
6605 16443 025404 LDB LWAM
6606 16444 004156 ADB M100
6607 16445 074076 TCB
6608 16446 005461 ADB FTMP2 EXCEEDED MAXIMUM
6609 16447 074112 SBM *+2 COMMON AREA?
6610 16450 064561 JMP FSCE4 YES
6611 16451 025461 LDB FTMP2 MOVE PROGRAM
6612 16452 035424 STB PBUFF POINTERS
6613 16453 035423 STB PBPTR
6614 16454 035426 STB PBPTO
6615 16455 164203 JMP ACTSA,I
6616*
6617* DIM STMT SYNTAX
6618*
6619 16456 062410 DIMS JSM ARRYS CHECK SYNTAX
6620 16457 164203 JMP ACTSA,I
6621*
6622* READ AND INPUT STATEMENT SYNTAX
6623*
6624 16460 160246 READS JSM VAROA,I RECORD VARIABLE OPERAND
6625 16461 160206 SYE13 JSM ERRA,I NONE FOUND
6626 16462 066463 JMP *+1

PAGE 143

SYNTAX

| | | | | | | |
|-------|------------------------------|--------|-------|-----|---------|------------------------------|
| 6627 | 16463 | 060504 | | JSM | COMCK | COMMA? |
| 6628 | 16464 | 065151 | | JMP | EOST | NO |
| 6629 | 16465 | 066460 | | JMP | READS | YES |
| 6630* | | | | | | |
| 6631* | RESTORE STMT SYNTAX | | | | | |
| 6632* | | | | | | |
| 6633 | 16466 | 061066 | RSTRS | JSM | GETSK | LINE # GIVEN? |
| 6634 | 16467 | 065150 | | JMP | ENDS+1 | NO |
| 6635 | 16470 | 060663 | | JSM | LNUM-1 | YES, GET LINE # |
| 6636 | 16471 | 065151 | | JMP | EOST | |
| 6637* | | | | | | |
| 6638* | REM STATEMENT SYNTAX | | | | | |
| 6639* | | | | | | |
| 6640 | 16472 | 020107 | REMS | LDA | B1000 | |
| 6641 | 16473 | 041552 | | IOR | RBUFF+1 | |
| 6642 | 16474 | 031552 | | STA | RBUFF+1 | SET ASCII BIT |
| 6643 | 16475 | 020031 | | LDA | EOL | SET STATEMENT TERMINATOR |
| 6644 | 16476 | 024012 | | LDB | .1 | SET TO STORE IN LOW HALF |
| 6645 | 16477 | 160253 | | JSM | CHRSA,I | RECORD REST TO STMT AS ASCII |
| 6646 | 16500 | 164203 | | JMP | ACTSA,I | |
| 6647* | | | | | | |
| 6648* | GET SELECT CODE AND FORMAT # | | | | | |
| 6649* | | | | | | |
| 6650 | 16501 | 045706 | IOHED | ISZ | SBPTR | |
| 6651 | 16502 | 060520 | | JSM | LPCKE-1 | DEMAND LEFT BRACKET |
| 6652 | 16503 | 160202 | | JSM | FSCA,I | GET SELECT CODE |
| 6653 | 16504 | 060510 | | JSM | COMCE | DEMAND COMMA |
| 6654 | 16505 | 061055 | | JSM | GNEXT | |
| 6655 | 16506 | 045706 | | ISZ | SBPTR | |
| 6656 | 16507 | 060463 | | JSM | SYMCI | LOOK FOR "*" |
| 6657 | 16510 | 012012 | | DEF | TIMES-1 | |
| 6658 | 16511 | 066514 | | JMP | *+3 | NOT FOUND |
| 6659 | 16512 | 045706 | | ISZ | SBPTR | |
| 6660 | 16513 | 065055 | | JMP | GNEXT | |
| 6661 | 16514 | 055706 | | DSZ | SBPTR | |
| 6662 | 16515 | 064663 | | JMP | LNUM-1 | |

PAGE 144

SYNTAX

6664*
 6665* SEARCH ALL TABLES
 6666* (A)=FIRST CHAR BADDR POINTS TO REST OF STRING
 6667* (B)=0 FOR STATEMENTS
 6668* =-1 FOR SYSTEM COMMANDS
 6669* =-2 FOR ARITHMETIC FUNCTIONS
 6670*
 6671 16516 004132 SALT ADB MI
 6672 16517 035473 STB TEMP SAVE RELATIVE ADDRESS OF TABLE
 6673 16520 025403 LDB MAW GET ADDRESS OF LAST
 6674 16521 062531 JSM SPAGE SEARCH TABLE ON R/W PAGE
 6675 16522 024131 LDB LROMA GET ADDRESS OF LAST ROM PAGE
 6676 16523 062531 SALT1 JSM SPAGE SEARCH TABLE IN ROM
 6677 16524 025717 LDB NUM
 6678 16525 014130 CPB FROMA ALL TABLES SEARCHED?
 6679 16526 170402 RET YES
 6680 16527 004160 ADB M1024
 6681 16530 066523 JMP SALT1 NO, SEARCH NEXT PAGE
 6682*
 6683* SEARCH TABLE ON A GIVEN PAGE
 6684* RET IF NOT FOUND
 6685* RET TO P+2 OF ROUTINE WHICH CALLED SALT IF FOUND
 6686* (A)= OP CODE
 6687* (B)= PAGE
 6688* (SBPTR,I)=SHIFTED OP CODE
 6689* (TEMP)=JUMP ADDRESS
 6690*
 6691 16531 035717 SPAGE STB NUM SAVE PAGE ADDRESS
 6692 16532 074537 LDB B,I READ LAST WORD OF PAGE
 6693 16533 074310 SZB TSRET RETURN IF ZERO
 6694 16534 025717 LDB NUM GET TABLE
 6695 16535 005473 ADB TEMP STARTING
 6696 16536 074437 ADB B,I ADDRESS
 6697 16537 074744 SBL 1 POINT TO FIRST CHARACTER
 6698 16540 060537 JSM STSRH SEARCH TABLE
 6699 16541 170402 TSRET NOT FOUND, RETURN
 6700 16542 070442 SAR 10
 6701 16543 025473 LDB TEMP
 6702 16544 014133 CPB M2
 6703 16545 050032 AND B17 GET 4 BIT CODE
 6704 16546 005717 ADB NUM FOR SYSTEM COMMANDS
 6705 16547 074437 ADB B,I GET TABLE
 6706 16550 070076 TCA ADDRESS
 6707 16551 074017 ADA B COMPUTE AND
 6708 16552 070417 ADA A,I STORE
 6709 16553 031473 STA TEMP ABSOLUTE
 6710 16554 125717 LDB NUM,I JUMP ADDRESS
 6711 16555 021466 LDA LOCL1 GET OP CODE FOR THIS PAGE
 6712 16556 055777 DSZ RSPTR RECALL OP CODE
 6713 16557 064357 JMP RETN2 RETURN TO ROUTINE
 WHICH CALLED SALT

EXECUTION

| | | | | | |
|--------------|----------------------------------|--------|-------|-----|---------|
| 6715* | | | | | |
| 6716* | EXECUTE GOTO AND GOSUB | | | | |
| 6717* | | | | | |
| 6718 | 16560 | 025421 | EGOSB | LDB | PRADD |
| 6719 | 16561 | 060563 | EGOTO | JSM | SLWST |
| 6720 | 16562 | 060771 | | JSM | GTOPP |
| 6721 | 16563 | 010064 | | CPA | INTFL |
| 6722 | 16564 | 066602 | | JMP | EGOS2 |
| 6723 | 16565 | 061217 | | JSM | FETCH |
| 6724 | 16566 | 160225 | | JSM | FLTRA,I |
| 6725 | 16567 | 066606 | | JMP | EGOS3 |
| 6726 | 16570 | 004154 | | ADB | M64 |
| 6727 | 16571 | 074653 | | SBP | EGOS3 |
| 6728 | 16572 | 004054 | | ADB | .63 |
| 6729 | 16573 | 074552 | | SBM | EGOS3 |
| 6730 | 16574 | 074744 | | SBL | 1 |
| 6731 | 16575 | 005472 | | ADB | TEMPS |
| 6732 | 16576 | 035472 | | STB | TEMPS |
| 6733 | 16577 | 074056 | | CMB | |
| 6734 | 16600 | 005421 | | ADB | PRADD |
| 6735 | 16601 | 074252 | | SBM | EGOS3 |
| 6736 | 16602 | 060567 | EGOS2 | JSM | UNSTK |
| 6737 | 16603 | 011421 | | CPA | PRADD |
| 6738 | 16604 | 045412 | | ISZ | LSTPT |
| 6739 | 16605 | 166610 | | JMP | EGSIA,I |
| 6740 | 16606 | 055412 | EGOS3 | DSZ | LSTPT |
| 6741 | 16607 | 164172 | | JMP | XEC4A,I |
| 6742 | 16610 | 010202 | EGSIA | DEF | EGOS1 |
| 6743* | 16614B IS LAST | | | | |
| 6744* | TAPE COMMAND SYNTAX FOLLOWS HERE | | | | |
| 6745* | | | | | |
| 6746 | END | | | | |
| * NO ERRORS* | | | | | |

CROSS-REFERENCE TABLE

| | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|
| SLBF | 0181 | 0308 | 1705 | 1856 | 1861 | 1900 | 2062 | 2542 | | | |
| A | 0000 | 0297 | 0304 | 0629 | 0664 | 0669 | 0671 | 0693 | 0695 | 0722 | 0725 |
| | 0754 | 0948 | 1004 | 1006 | 1011 | 1050 | 1423 | 2072 | 2192 | 2209 | 2220 |
| | 2923 | 2930 | 2945 | 3148 | 3163 | 3430 | 3543 | 3545 | 3551 | 3587 | 3599 |
| | 3623 | 3669 | 3670 | 3676 | 3859 | 4172 | 4584 | 4594 | 4626 | 4684 | 4701 |
| | 4703 | 4706 | 5083 | 5201 | 5211 | 5304 | 5321 | 5515 | 5537 | 5545 | 5595 |
| | 5658 | 5853 | 5882 | 5952 | 5962 | 6020 | 6056 | 6176 | 6184 | 6309 | 6708 |
| ACCS1 | 2876 | 2867 | 2873 | | | | | | | | |
| ACCS2 | 2889 | 2883 | 2894 | 2903 | | | | | | | |
| ACCS3 | 2881 | 2875 | | | | | | | | | |
| ACCS4 | 2900 | | 2872 | 2885 | | | | | | | |
| ACTKY | 2861 | 2671 | | | | | | | | | |
| ACTSA | 0174 | 0862 | 6615 | 6620 | 6646 | | | | | | |
| ACTST | 2854 | 0174 | | | | | | | | | |
| ADRSS | 1112 | | | | | | | | | | |
| AERR | 2758 | 0176 | | | | | | | | | |
| AERRA | 0176 | 0352 | 0967 | 0972 | | | | | | | |
| AIEX1 | 2249 | 2113 | | | | | | | | | |
| AIEX2 | 2250 | 2120 | | | | | | | | | |
| ALCLI | 2246 | 2142 | | | | | | | | | |
| ALNUM | 2554 | | | | | | | | | | |
| ANDOP | 6125 | | | | | | | | | | |
| ARI | 1198 | 0182 | 0355 | 1207 | 3989 | 4001 | | | | | |
| ARIA | 0182 | 0324 | 0363 | | | | | | | | |
| ARIE | 1207 | 0940 | 0945 | | | | | | | | |
| AR2 | 1200 | 0183 | 1208 | 2135 | 2141 | 2246 | 2246 | 3990 | 4002 | 4091 | 4114 |
| | 4147 | 4151 | 4230 | 5145 | 5314 | | | | | | |
| AR2A | 0183 | 0325 | 0364 | 0956 | 0991 | 5474 | 5689 | 5847 | 5907 | | |
| AR2E | 1208 | 0942 | 0954 | 0961 | 0979 | 0982 | 0984 | | | | |
| ARBAS | 5741 | 5542 | | | | | | | | | |
| ARREI | 6593 | 0251 | 6581 | | | | | | | | |
| ARRY1 | 6589 | | | | | | | | | | |
| ARRY5 | 6575 | 6592 | 6604 | 6619 | | | | | | | |
| ARW | 1214 | 1502 | 3728 | 3733 | 3744 | | | | | | |
| ARYAD | 1188 | 6357 | 6358 | 6374 | 6417 | 6425 | 6429 | 6461 | 6477 | | |
| ASSOP | 6109 | 6263 | 6488 | 6521 | | | | | | | |
| ASTR1 | 2247 | 2108 | | | | | | | | | |
| ASTR2 | 2248 | 2117 | | | | | | | | | |
| AUTO | 2400 | 2288 | | | | | | | | | |
| B | 0000 | 0309 | 0318 | 0319 | 0376 | 0399 | 0409 | 0515 | 0691 | 0718 | 0721 |
| | 0732 | 0752 | 0755 | 0798 | 0805 | 0826 | 0915 | 0939 | 0947 | 1054 | 1062 |
| | 1065 | 1385 | 1500 | 1711 | 1807 | 2085 | 2094 | 2103 | 2111 | 2116 | 2121 |
| | 2124 | 2129 | 2136 | 2179 | 2191 | 2197 | 2208 | 2219 | 2233 | 2237 | 2489 |
| | 2490 | 2495 | 2768 | 2890 | 2917 | 2926 | 2931 | 2934 | 3107 | 3110 | 3121 |
| | 3125 | 3138 | 3144 | 3173 | 3191 | 3222 | 3225 | 3236 | 3305 | 3341 | 3350 |
| | 3358 | 3369 | 3372 | 3378 | 3382 | 3383 | 3394 | 3412 | 3415 | 3432 | 3457 |
| | 3476 | 3697 | 3730 | 3802 | 3841 | 3921 | 4203 | 4500 | 4592 | 4636 | 4666 |
| | 5029 | 5030 | 5169 | 5246 | 5308 | 5492 | 5497 | 5505 | 5523 | 5555 | 5562 |
| | 5564 | 5575 | 5585 | 5587 | 5637 | 5638 | 5660 | 5807 | 5811 | 5827 | 5955 |
| | 6015 | 6027 | 6187 | 6449 | 6692 | 6696 | 6705 | 6707 | | | |
| B100 | 0095 | 1324 | 1340 | 1448 | 2139 | 3891 | | | | | |
| B1000 | 0102 | 3571 | 3795 | 6640 | | | | | | | |
| B10K | 0114 | 0115 | 1245 | 2243 | | | | | | | |
| B12K | 0116 | 0117 | 0206 | | | | | | | | |
| B133 | 0096 | 6330 | | | | | | | | | |
| B137 | 2049 | 1821 | | | | | | | | | |

PAGE 149

CROSS-REFERENCE TABLE

| | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|
| B140 | 0097 | 1051 | 1052 | 1622 | | | | | | | | |
| B1400 | 0103 | 0204 | | | | | | | | | | |
| B152 | 1623 | 1312 | | | | | | | | | | |
| B17 | 0083 | 5455 | 6703 | | | | | | | | | |
| B170K | 0121 | 1837 | 2539 | 2580 | 2786 | | | | | | | |
| B1740 | 0104 | 0203 | 5973 | 6029 | | | | | | | | |
| B176 | 2048 | 1827 | | | | | | | | | | |
| B1760 | 0105 | 0158 | 3336 | | | | | | | | | |
| B177 | 0098 | 0694 | 1809 | 1817 | 4578 | 4632 | 4997 | | | | | |
| B1774 | 0106 | | | | | | | | | | | |
| B1775 | 0107 | | | | | | | | | | | |
| B1777 | 0108 | 3339 | 3344 | 3471 | 3511 | 3943 | 5329 | 5659 | | | | |
| B20 | 0084 | 5005 | 5533 | | | | | | | | | |
| B200 | 2044 | 1926 | | | | | | | | | | |
| B22 | 0085 | 1282 | 1308 | | | | | | | | | |
| B23 | 0086 | 1310 | | | | | | | | | | |
| B2400 | 0110 | | | | | | | | | | | |
| B26 | 0087 | 1986 | | | | | | | | | | |
| B27 | 0088 | 1984 | | | | | | | | | | |
| B2K | 1244 | | | | | | | | | | | |
| B3000 | 0111 | | | | | | | | | | | |
| B33 | 2047 | 1417 | 1825 | | | | | | | | | |
| B36 | 0089 | 3218 | 3260 | 3897 | 4574 | 6476 | | | | | | |
| B37 | 0090 | 0456 | 1063 | 3259 | 3263 | 3600 | 3689 | 4573 | 4625 | 5462 | 5463 | |
| | 5498 | 5502 | 5524 | 5528 | 6183 | 6511 | | | | | | |
| B377 | 0099 | 1953 | 2104 | 3295 | 4501 | 5808 | | | | | | |
| B400 | 0100 | 4000 | | | | | | | | | | |
| B4000 | 0112 | 0162 | 1803 | 4595 | | | | | | | | |
| B42 | 0091 | 2370 | 2717 | 2726 | 4865 | | | | | | | |
| B43 | 2429 | 2410 | | | | | | | | | | |
| B44 | 0092 | 1243 | 6305 | 6462 | | | | | | | | |
| B4400 | 3244 | 3206 | | | | | | | | | | |
| B52 | 0093 | 1652 | 2247 | 2685 | 3770 | 3787 | | | | | | |
| B57 | 1624 | 1418 | | | | | | | | | | |
| B60 | 1241 | 4102 | 4193 | 4195 | 4210 | | | | | | | |
| B6K | 0113 | 0167 | | | | | | | | | | |
| B72 | 0094 | | | | | | | | | | | |
| B777 | 0101 | 3598 | | | | | | | | | | |
| BADD | 2377 | 2364 | | | | | | | | | | |
| BADDR | 1076 | 0311 | 0530 | 0539 | 0572 | 0777 | 1285 | 1289 | 1302 | 1511 | 1515 | |
| | 1519 | 1542 | 1707 | 1792 | 1833 | 1835 | 1842 | 1855 | 1858 | 1876 | 1885 | |
| | 1914 | 2017 | 2027 | 2063 | 2346 | 2412 | 2541 | 2563 | 2642 | 2644 | 2709 | |
| | 2749 | 2801 | 2802 | 4386 | 4393 | 4412 | 4415 | 4509 | 4521 | 4528 | 4721 | |
| | 4728 | 4944 | 5402 | 5422 | 6165 | 6171 | 6242 | 6529 | | | | |
| BADRP | 0204 | 0716 | 0750 | 0766 | | | | | | | | |
| BASE | 4629 | | | | | | | | | | | |
| BEEB | 1625 | 1422 | | | | | | | | | | |
| BEND | 0198 | 0199 | 1401 | 1709 | | | | | | | | |
| BERR | 5267 | 0282 | 5258 | 5273 | | | | | | | | |
| BHSTP | 0471 | 0486 | 3544 | 5134 | 5208 | 5471 | 5506 | 5549 | 5731 | 5899 | 6004 | |
| BINO1 | 5969 | 5974 | | | | | | | | | | |
| BINO2 | 5975 | 5971 | | | | | | | | | | |
| BINOP | 5972 | 5750 | 5840 | 5860 | 5866 | 5872 | 5905 | 5950 | 5960 | | | |
| BK1 | 1866 | 1860 | | | | | | | | | | |
| BK2 | 1861 | 1854 | | | | | | | | | | |
| BKPTA | 2045 | 1349 | 1940 | 1949 | 1967 | | | | | | | |

265

PAGE 150

CROSS-REFERENCE TABLE

| | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|
| BKPTR | 1230 | 1275 | 1347 | 1509 | 1516 | 1535 | 1704 | 1853 | 1859 | 1865 | 1870 |
| | 1872 | 1875 | 1915 | 1916 | 1938 | 1943 | 1946 | 1951 | 1957 | 1969 | 2045 |
| BKSP | 1853 | 1309 | | | | | | | | | |
| BLANK | 1173 | 0768 | 2555 | | | | | | | | |
| BLNKA | 4467 | 4463 | | | | | | | | | |
| C5 | 0015 | | | | | | | | | | |
| C6 | 0011 | 0114 | | | | | | | | | |
| C7 | 0022 | 0100 | | | | | | | | | |
| CALEX | 5056 | 2838 | 5009 | | | | | | | | |
| CALLK | 1976 | 1277 | 1306 | | | | | | | | |
| CALSN | 2682 | 2582 | | | | | | | | | |
| CALX2 | 5071 | | | | | | | | | | |
| CALXA | 2838 | 2828 | | | | | | | | | |
| CCNT | 1229 | 1274 | 1438 | 1454 | 1917 | 1927 | 1930 | 1934 | 2091 | 2107 | |
| CHRS1 | 4890 | 4900 | 4901 | | | | | | | | |
| CHRS2 | 4902 | 4897 | | | | | | | | | |
| CHRS3 | 4905 | 4893 | 4895 | | | | | | | | |
| CHRSA | 0219 | 2670 | 6645 | | | | | | | | |
| CHRST | 4889 | 0219 | 4871 | | | | | | | | |
| CKEO1 | 4554 | 4550 | | | | | | | | | |
| CKEOL | 4546 | 4266 | 4335 | 4441 | 4472 | 4529 | | | | | |
| CKRCD | 2587 | 2573 | | | | | | | | | |
| CLAR2 | 0956 | 0969 | 0976 | 3992 | | | | | | | |
| CLE1 | 1711 | 1714 | | | | | | | | | |
| CLEAR | 1703 | 0160 | 1294 | 1361 | 1366 | 1393 | 1437 | | | | |
| CLERA | 0160 | 1026 | 2419 | 2520 | 2560 | 2784 | 3739 | 4293 | 4333 | 4419 | 5102 |
| | 5140 | 5393 | 5427 | 5434 | 5437 | | | | | | |
| CLINE | 1101 | 0792 | 1429 | 1465 | 1553 | 2623 | 3768 | | | | |
| CLPIA | 0217 | 2195 | | | | | | | | | |
| CLPRI | 2915 | 0217 | 2920 | 2979 | | | | | | | |
| CLPR2 | 2921 | 2916 | 2953 | | | | | | | | |
| CLPRG | 2908 | 2847 | 2887 | | | | | | | | |
| CLRST | 1687 | 1360 | 1680 | | | | | | | | |
| CMDI3 | 2296 | 6567 | | | | | | | | | |
| CMERR | 3134 | 0268 | | | | | | | | | |
| CMND | 2619 | 2610 | | | | | | | | | |
| CMNDF | 2622 | | | | | | | | | | |
| COLON | 1242 | 5109 | | | | | | | | | |
| COMCE | 0419 | 0248 | 2362 | 2392 | 2465 | 6591 | 6653 | | | | |
| COMCK | 0414 | 0419 | 2336 | 6627 | | | | | | | |
| COML | 1209 | 3053 | 3124 | 3126 | 3132 | | | | | | |
| COMMA | 6104 | 2720 | | | | | | | | | |
| COMP1 | 3649 | 3643 | | | | | | | | | |
| COMPL | 3640 | 2989 | 3651 | | | | | | | | |
| COMPR | 5905 | 5912 | 5918 | 5924 | 5930 | 5936 | 5942 | | | | |
| COMS | 6597 | 4800 | | | | | | | | | |
| COMTA | 1628 | 1362 | | | | | | | | | |
| CONS3 | 0872 | 0870 | | | | | | | | | |
| CONST | 0867 | 2333 | 5413 | | | | | | | | |
| CONT | 1382 | 2283 | | | | | | | | | |
| CONT1 | 1388 | 1383 | | | | | | | | | |
| CONTK | 1386 | 1384 | 2149 | | | | | | | | |
| CPLAC | 1116 | 2504 | 4305 | 4947 | | | | | | | |
| CR1 | 1040 | 1028 | 1804 | | | | | | | | |
| CRC | 1670 | 1414 | | | | | | | | | |
| CRCI | 1678 | 1671 | 4768 | | | | | | | | |

PAGE 151

CROSS-REFERENCE TABLE

| | | | | | | | | | | | | |
|--------|------|------|------|------|------|------|------|------|------|------|------|--|
| CRCIA | 4768 | 3757 | | | | | | | | | | |
| CRC2 | 1676 | 1673 | | | | | | | | | | |
| CRCHK | 1672 | 0228 | | | | | | | | | | |
| CRCKA | 0228 | 2783 | 2808 | 5071 | 5101 | | | | | | | |
| CRLF | 1027 | | | | | | | | | | | |
| CRLF1 | 1029 | 1681 | 2780 | 3732 | 3754 | 4347 | 4421 | 4454 | 5025 | 5122 | | |
| CRTN | 0037 | 0038 | 0039 | 1029 | 1345 | 1654 | 1819 | | | | | |
| CTERM | 1683 | 1675 | | | | | | | | | | |
| CTYPE | 1115 | 2505 | 4306 | 4939 | | | | | | | | |
| CXEC4 | 5437 | 5420 | | | | | | | | | | |
| DATA2 | 2336 | | | | | | | | | | | |
| DATAS | 2332 | 2338 | 4788 | | | | | | | | | |
| DATCD | 5386 | 5379 | | | | | | | | | | |
| DCMF1 | 3689 | | | | | | | | | | | |
| DCMF2 | 3693 | 3699 | | | | | | | | | | |
| DCMPI | 3675 | 3681 | | | | | | | | | | |
| DCMP2 | 3683 | 3674 | 3680 | | | | | | | | | |
| DCMP3 | 3679 | 3675 | | | | | | | | | | |
| DCMP4 | 3702 | 3667 | | | | | | | | | | |
| DCMP5 | 3666 | 3686 | | | | | | | | | | |
| DCMP6 | 3672 | 3707 | | | | | | | | | | |
| DCOM1 | 3661 | | | | | | | | | | | |
| DCOMA | 0210 | 0567 | 2167 | 2188 | 2842 | 2861 | 2977 | | | | | |
| DCOMF | 3688 | 3664 | 3685 | | | | | | | | | |
| DCOMP | 3655 | 0210 | 2968 | | | | | | | | | |
| DEFOP | 3532 | 3467 | 3507 | | | | | | | | | |
| DEFS | 6507 | 4799 | | | | | | | | | | |
| DEL1 | 1427 | 1280 | 1571 | 1574 | | | | | | | | |
| DEL2 | 1426 | 1428 | | | | | | | | | | |
| DELCl | 1518 | 1527 | | | | | | | | | | |
| DELET | 2975 | 2284 | | | | | | | | | | |
| DELIN | 2050 | 1279 | | | | | | | | | | |
| DELSA | 0175 | 1433 | | | | | | | | | | |
| DELST | 2842 | 0175 | 2596 | | | | | | | | | |
| DFLAG | 1175 | 2607 | 2944 | 2952 | 6322 | 6325 | 6359 | 6387 | 6575 | | | |
| DIGCK | 0650 | 0597 | 0599 | 2343 | 2458 | 4942 | 6307 | | | | | |
| DIGIT | 1233 | 4025 | 4035 | 4059 | 4124 | | | | | | | |
| DIMS | 6619 | 4801 | 6602 | | | | | | | | | |
| DINCT | 6435 | 4606 | 6409 | 6412 | 6418 | 6419 | | | | | | |
| DISP | 6544 | 5039 | | | | | | | | | | |
| DISPA | 5039 | 5034 | | | | | | | | | | |
| DIV | 6114 | | | | | | | | | | | |
| DLIMIT | 0648 | 0650 | | | | | | | | | | |
| DNARW | 1473 | 1566 | | | | | | | | | | |
| DOLAR | 1243 | 3896 | 4073 | | | | | | | | | |
| DPERR | 0043 | | | | | | | | | | | |
| DRQST | 1102 | 0928 | 2069 | 2525 | 2572 | 2576 | 2820 | 2822 | 3043 | 3737 | 5137 | |
| | 5400 | 5412 | 5416 | | | | | | | | | |
| DSTRA | 5698 | 5614 | | | | | | | | | | |
| DSTRT | 1088 | 3657 | 5365 | 5383 | 5449 | 5645 | 5698 | | | | | |
| E | 0080 | 4089 | | | | | | | | | | |
| E1 | 0235 | 2836 | | | | | | | | | | |
| E10 | 0244 | | | | | | | | | | | |
| E15 | 0249 | | | | | | | | | | | |
| E20 | 0254 | | | | | | | | | | | |
| E25 | 0259 | 4911 | 4912 | 6580 | | | | | | | | |

[illegible]

CROSS-REFERENCE TABLE

| | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|
| ENTE6 | 3485 | 3480 | | | | | | | | | | |
| ENTEA | 0218 | 5197 | 5496 | | | | | | | | | |
| ENTER | 3420 | 0218 | | | | | | | | | | |
| ENTFA | 0189 | 5591 | | | | | | | | | | |
| ENTFN | 3449 | 0189 | | | | | | | | | | |
| ENTRY | 2483 | 2961 | | | | | | | | | | |
| EOL | 0038 | 0779 | 0861 | 1824 | 1831 | 1846 | 1928 | 2462 | 2595 | 4892 | 6249 | |
| | 6589 | 6643 | | | | | | | | | | |
| EOLCK | 0861 | 0856 | 2361 | 2391 | 2394 | 2659 | 2731 | 2751 | 6495 | 6519 | | |
| EOLL | 1399 | 1346 | | | | | | | | | | |
| EOR | 5950 | 5772 | | | | | | | | | | |
| EOR2 | 5954 | 5964 | | | | | | | | | | |
| EOST | 0856 | 2337 | 2691 | 2725 | 4913 | 4922 | 6503 | 6528 | 6628 | 6636 | | |
| EPI | 6046 | 6044 | | | | | | | | | | |
| EPRIN | 4326 | 6547 | | | | | | | | | | |
| EPWR | 5872 | 5763 | | | | | | | | | | |
| EQUAL | 6119 | | | | | | | | | | | |
| ER3 | 5176 | 0281 | 5179 | 5184 | 5188 | | | | | | | |
| ER6 | 5823 | 0276 | 5795 | 5801 | 5816 | | | | | | | |
| ER60 | 5377 | 0283 | 5381 | | | | | | | | | |
| ERB | 6017 | 0274 | | | | | | | | | | |
| ERAD | 4927 | 4931 | 4933 | 6541 | | | | | | | | |
| ERBS | 2836 | 2766 | 2771 | | | | | | | | | |
| EREAD | 4968 | 4974 | 6548 | | | | | | | | | |
| EREC | 1437 | 1573 | | | | | | | | | | |
| ERES | 6048 | 6045 | | | | | | | | | | |
| ERET | 2831 | 2825 | 2826 | 2827 | | | | | | | | |
| ERN1 | 2818 | 2811 | | | | | | | | | | |
| ERR1 | 2772 | 2760 | | | | | | | | | | |
| ERR2 | 2781 | 2774 | 2776 | | | | | | | | | |
| ERR3 | 2801 | 2795 | | | | | | | | | | |
| ERR4 | 2809 | | | | | | | | | | | |
| ERR5 | 2820 | 2813 | | | | | | | | | | |
| ERRA | 0177 | 0420 | 0433 | 0448 | 0447 | 0542 | 0618 | 0857 | 0874 | 2084 | 2334 | |
| | 2358 | 2389 | 2667 | 2692 | 2713 | 3423 | 4426 | 4665 | 4683 | 4867 | 4872 | |
| | 4920 | 5018 | 5168 | 5176 | 5267 | 5377 | 5823 | 6017 | 6217 | 6545 | 6472 | |
| | 6486 | 6493 | 6498 | 6517 | 6522 | 6593 | 6600 | 6625 | | | | |
| ERRM | 2835 | 2833 | 2833 | 2834 | 2834 | | | | | | | |
| ERROR | 2764 | 0177 | 2617 | 2658 | 2837 | | | | | | | |
| ERSTR | 5444 | 6546 | | | | | | | | | | |
| ERTN1 | 4990 | 5189 | 5322 | | | | | | | | | |
| ERTRN | 5177 | 6553 | | | | | | | | | | |
| ESCMA | 5793 | 5756 | | | | | | | | | | |
| ESTAC | 5128 | 4984 | 5095 | | | | | | | | | |
| ESTOP | 5070 | 5052 | 5123 | 6551 | | | | | | | | |
| ESTR | 5781 | 5757 | | | | | | | | | | |
| ESTR1 | 5788 | 5786 | | | | | | | | | | |
| ESYM1 | 3412 | 3446 | | | | | | | | | | |
| ESYMT | 3407 | 3113 | 3434 | | | | | | | | | |
| ET1 | 4419 | 4417 | | | | | | | | | | |
| ETAB | 4405 | 4283 | | | | | | | | | | |
| EUMIN | 5878 | 5768 | | | | | | | | | | |
| EVARG | 5455 | 5570 | 5609 | | | | | | | | | |
| EVMLF | 5702 | 5665 | | | | | | | | | | |
| EW1 | 4760 | 4761 | 4763 | | | | | | | | | |
| EW2 | 4733 | 4737 | 4764 | | | | | | | | | |

CROSS-REFERENCE TABLE

| | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|
| EWAIT | 4758 | 6550 | | | | | | | | | | |
| EWRI1 | 4329 | 4440 | 4459 | | | | | | | | | |
| EWRI2 | 4438 | 6538 | | | | | | | | | | |
| EXP | 1186 | 1187 | | | | | | | | | | |
| EXPOP | 0118 | 6259 | | | | | | | | | | |
| EXPS | 6115 | | | | | | | | | | | |
| EXPTB | 1580 | 1356 | | | | | | | | | | |
| F | 0081 | 3912 | | | | | | | | | | |
| FADD | 2375 | 2351 | | | | | | | | | | |
| FALSE | 5944 | 5914 | 5926 | 5932 | 5963 | | | | | | | |
| FDAAA | 0222 | 4969 | | | | | | | | | | |
| FDATA | 5361 | 0222 | 5384 | | | | | | | | | |
| FETCH | 0911 | 4288 | 4407 | 4480 | 4663 | 4758 | 5144 | 5229 | 5239 | 5787 | 5793 | |
| | 5799 | 6723 | | | | | | | | | | |
| FILL | 4391 | 0201 | 4320 | | | | | | | | | |
| FILL1 | 4380 | 4391 | 4392 | | | | | | | | | |
| FILL2 | 4398 | 4388 | 4414 | | | | | | | | | |
| FILLA | 0201 | | | | | | | | | | | |
| FIRST | 1152 | 1700 | 4384 | 4527 | 4719 | 4726 | 4729 | | | | | |
| FIX | 1061 | 0320 | 5226 | 5262 | 5288 | 5297 | 5327 | | | | | |
| FIXED | 4939 | 2281 | 4943 | | | | | | | | | |
| FLGBT | 0073 | 1269 | 3002 | 3172 | 3431 | 4422 | 4952 | 4995 | 5601 | 6016 | | |
| FLOAT | 1171 | 3829 | 4041 | 4430 | 4484 | 4732 | 4736 | | | | | |
| FLT | 4938 | 2280 | | | | | | | | | | |
| FLT1 | 4066 | 4056 | | | | | | | | | | |
| FLT10 | 4097 | 4094 | | | | | | | | | | |
| FLT11 | 4064 | | | | | | | | | | | |
| FLT2 | 4077 | 4055 | 4065 | | | | | | | | | |
| FLT3 | 4073 | | | | | | | | | | | |
| FLT4 | 4076 | 4072 | | | | | | | | | | |
| FLT5 | 4080 | 4062 | | | | | | | | | | |
| FLT6 | 4086 | 4088 | | | | | | | | | | |
| FLT7 | 4063 | 4058 | | | | | | | | | | |
| FLT8 | 4089 | 4083 | | | | | | | | | | |
| FLT9 | 4104 | 4101 | | | | | | | | | | |
| FLTFA | 0193 | 4408 | 4759 | | | | | | | | | |
| FLTRA | 0192 | 0348 | 4498 | 4664 | 5794 | 5800 | 6724 | | | | | |
| FLTSA | 0195 | 0361 | | | | | | | | | | |
| FMTB | 2364 | 2353 | | | | | | | | | | |
| FMTC | 2361 | 2369 | 2374 | | | | | | | | | |
| FMTD | 2359 | | | | | | | | | | | |
| FMTER | 2358 | 0266 | 2372 | | | | | | | | | |
| FMTF | 2350 | 2344 | | | | | | | | | | |
| FMTOP | 4712 | 4438 | | | | | | | | | | |
| FMTQT | 2370 | 2366 | | | | | | | | | | |
| FMTS | 2342 | 2363 | 4782 | | | | | | | | | |
| FMX80 | 5593 | 5602 | | | | | | | | | | |
| FMX81 | 5647 | 5630 | | | | | | | | | | |
| FMX82 | 5631 | 5625 | 5629 | | | | | | | | | |
| FMX83 | 5644 | 5634 | | | | | | | | | | |
| FMX84 | 5633 | 5641 | | | | | | | | | | |
| FMX9 | 5668 | 5714 | | | | | | | | | | |
| FN | 0148 | 2134 | 6450 | | | | | | | | | |
| FNCK | 6447 | 6163 | 6509 | | | | | | | | | |
| FND | 0639 | 0665 | 0670 | 1499 | 2439 | 2443 | | | | | | |
| FND1A | 0224 | | | | | | | | | | | |

CROSS-REFERENCE TABLE

[illegible]

[illegible]

CROSS-REFERENCE TABLE

[illegible]

PAGE 158

CROSS-REFERENCE TABLE

| | | | | | | | | | | | |
|--------|------|------|------|------|------|------|------|------|------|------|------|
| LDEF | 0920 | 2145 | 2151 | 2159 | 2187 | 2513 | 2897 | 2967 | 5050 | 5631 | |
| LEFT | 1885 | 1353 | 1987 | | | | | | | | |
| LETCK | 0657 | 0845 | 1332 | 2645 | 6158 | | | | | | |
| LETS | 2690 | 4781 | | | | | | | | | |
| LETSN | 2698 | 2648 | 2690 | | | | | | | | |
| LIMIT | 0660 | 0651 | 3856 | | | | | | | | |
| LIST | 3728 | | | | | | | | | | |
| LISTI | 3734 | 2043 | | | | | | | | | |
| LISTA | 0167 | 2418 | | | | | | | | | |
| LISTC | 2409 | 2290 | | | | | | | | | |
| LISTR | 2043 | 1504 | | | | | | | | | |
| LISTS | 3762 | 3743 | | | | | | | | | |
| LLIMIT | 0649 | 0657 | | | | | | | | | |
| LLL | 3783 | 3769 | | | | | | | | | |
| LMSG | 2834 | 2797 | | | | | | | | | |
| LN72 | 4770 | 4361 | | | | | | | | | |
| LNFD | 0072 | 1035 | 1823 | | | | | | | | |
| LNINC | 1100 | 1281 | 1292 | 1295 | 1298 | 1427 | 1432 | 2404 | 2475 | 2518 | 2556 |
| | 2624 | 2991 | 3017 | 3023 | 3750 | 5073 | | | | | |
| LNUM | 0573 | 2383 | 2390 | 2425 | 2460 | 2466 | 4921 | 6635 | 6662 | | |
| LOCL1 | 1130 | 0395 | 0403 | 0455 | 0511 | 0660 | 0666 | 0690 | 0751 | 0759 | 1014 |
| | 1017 | 1205 | 2086 | 2101 | 3402 | 3466 | 3541 | 3549 | 3566 | 3797 | 4253 |
| | 4470 | 4482 | 4486 | 4487 | 4488 | 4489 | 4493 | 4620 | 4631 | 4639 | 4652 |
| | 5028 | 5117 | 5154 | 5155 | 5158 | 5349 | 5563 | 5566 | 5728 | 5733 | 6711 |
| LOCL2 | 1131 | 0396 | 0401 | 0405 | 0717 | 0729 | 1206 | 1659 | 1830 | 3403 | 3918 |
| | 3922 | 4633 | 4649 | 4655 | 5726 | 5729 | | | | | |
| LPARN | 6123 | | | | | | | | | | |
| LPCK | 0425 | 0432 | 6200 | 6353 | | | | | | | |
| LPCKE | 0432 | 6514 | 6651 | | | | | | | | |
| LPOP | 0120 | 0428 | | | | | | | | | |
| LPTR | 1627 | | | | | | | | | | |
| LROMA | 0123 | 0517 | 6675 | | | | | | | | |
| LSS | 6117 | | | | | | | | | | |
| LSSEQ | 6128 | | | | | | | | | | |
| LSTAK | 1084 | 3181 | 3183 | 3186 | 3187 | 3188 | 3200 | 3209 | 3221 | 3224 | 3227 |
| | 5064 | 5089 | 5132 | 5136 | 5991 | 5996 | | | | | |
| LSTC1 | 2413 | 2428 | | | | | | | | | |
| LSTC2 | 2425 | 2411 | | | | | | | | | |
| LSTPT | 1086 | 0474 | 0484 | 0485 | 0491 | 0492 | 0563 | 2420 | 3054 | 3164 | 3165 |
| | 3171 | 3180 | 3232 | 4506 | 4534 | 4535 | 5059 | 5066 | 5078 | 5088 | 5128 |
| | 5133 | 5177 | 5178 | 5527 | 5539 | 5540 | 5573 | 5574 | 5580 | 5581 | 5611 |
| | 5613 | 5669 | 5672 | 5676 | 5682 | 5683 | 5685 | 5686 | 5707 | 5709 | 5727 |
| | 5730 | 5732 | 5783 | 5798 | 5821 | 5824 | 5835 | 5894 | 5997 | 6001 | 6437 |
| | 6738 | 6740 | | | | | | | | | |
| LTADD | 2678 | 2545 | | | | | | | | | |
| LTR | 0844 | 6301 | 6453 | 6507 | | | | | | | |
| LTRUE | 5938 | 5943 | 5953 | | | | | | | | |
| LWAM | 1080 | 0552 | 2498 | 6605 | | | | | | | |
| M1 | 0124 | 0296 | 0562 | 0825 | 0832 | 0883 | 1010 | 1064 | 1408 | 1464 | 1531 |
| | 1793 | 1954 | 1960 | 1980 | 2193 | 2497 | 2535 | 2604 | 2619 | 2673 | 2722 |
| | 2734 | 2810 | 2824 | 3004 | 3055 | 3068 | 3072 | 3095 | 3097 | 3102 | 3106 |
| | 3136 | 3190 | 3201 | 3235 | 3443 | 3463 | 3609 | 3695 | 3698 | 3749 | 3798 |
| | 3801 | 3828 | 3931 | 4057 | 4060 | 4068 | 4136 | 4171 | 4285 | 4302 | 4611 |
| | 4617 | 4628 | 4637 | 5138 | 5499 | 5554 | 5594 | 5603 | 5692 | 5796 | 5802 |
| | 5806 | 6152 | 6246 | 6253 | 6364 | 6366 | 6377 | 6473 | 6671 | | |
| M10 | 0131 | 1319 | 1795 | 4100 | 4161 | 5202 | 5675 | | | | |

CROSS-REFERENCE TABLE

| | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|
| MDIM | 3289 | 0200 | 3123 | 3211 | | | | | | | | |
| MDIM1 | 3307 | 3292 | | | | | | | | | | |
| MDIM2 | 3305 | 3291 | 3293 | 3303 | 3310 | | | | | | | |
| MDIMA | 0208 | 6585 | | | | | | | | | | |
| MDMA | 3710 | 3074 | | | | | | | | | | |
| MER10 | 3198 | 0272 | | | | | | | | | | |
| MER11 | 5018 | 0284 | | | | | | | | | | |
| MER5 | 3108 | 0269 | 3088 | | | | | | | | | |
| MER8 | 0477 | 0479 | 4523 | | | | | | | | | |
| MER9 | 3304 | 0273 | | | | | | | | | | |
| MERR | 2837 | 3108 | 3134 | 3198 | 3304 | 3368 | 3385 | | | | | |
| MINIT | 0552 | 2514 | 2829 | 3040 | | | | | | | | |
| MINS | 6112 | | | | | | | | | | | |
| MINUS | 1239 | 4095 | 4232 | | | | | | | | | |
| MISOF | 2389 | 0259 | | | | | | | | | | |
| ML2L | 3089 | 3081 | | | | | | | | | | |
| MLOP2 | 3149 | 3066 | 3071 | 3089 | 3111 | 3117 | 3151 | | | | | |
| MLOP5 | 3084 | 3079 | | | | | | | | | | |
| MLOP7 | 3073 | | | | | | | | | | | |
| MLOP8 | 3092 | 3080 | | | | | | | | | | |
| MLOP9 | 3113 | 3085 | 3105 | | | | | | | | | |
| MLP71 | 3154 | 3077 | | | | | | | | | | |
| MLP72 | 3075 | 3157 | | | | | | | | | | |
| MO1 | 4753 | 4745 | 4746 | | | | | | | | | |
| MO2 | 4750 | 4752 | | | | | | | | | | |
| MODE | 1090 | 1270 | 1315 | 1394 | 2434 | 2577 | 2583 | 2587 | 2613 | 2625 | 2655 | |
| | 2660 | 2781 | 2819 | 2823 | 2856 | 2858 | 3056 | 4052 | 4150 | 4303 | 4329 | |
| | 4514 | 4517 | 4520 | 4985 | 5007 | 5035 | 5148 | 5648 | | | | |
| MOUT | 4745 | 4075 | 4076 | 4177 | 4194 | 4211 | 4399 | | | | | |
| MOUTA | 1626 | 1364 | | | | | | | | | | |
| MOVCI | 3549 | 3554 | | | | | | | | | | |
| MOVCI | 3555 | 3550 | | | | | | | | | | |
| MOVCK | 3539 | 3214 | 3409 | 3427 | 3441 | | | | | | | |
| MOVST | 5726 | 5616 | 5704 | | | | | | | | | |
| MREM | 3713 | 3567 | | | | | | | | | | |
| MSFLG | 1174 | 2602 | 2698 | 6204 | 6213 | 6222 | 6255 | 6399 | 6407 | 6421 | 6577 | |
| | 6583 | | | | | | | | | | | |
| MSK65 | 1622 | 1316 | 1327 | | | | | | | | | |
| MSKIP | 3060 | 3065 | 3150 | | | | | | | | | |
| MSLPT | 4769 | 4394 | 4413 | | | | | | | | | |
| MSTND | 3067 | 3152 | | | | | | | | | | |
| MVTOH | 0821 | 0828 | 1406 | 2172 | 2902 | 5220 | 5681 | 6005 | | | | |
| N | 4771 | 3914 | | | | | | | | | | |
| N1 | 4150 | 4143 | 4146 | 4148 | | | | | | | | |
| N2 | 4137 | 4131 | | | | | | | | | | |
| N4 | 4145 | | | | | | | | | | | |
| N6 | 4147 | | | | | | | | | | | |
| N7 | 4164 | 4158 | | | | | | | | | | |
| NCALC | 2617 | 0237 | | | | | | | | | | |
| NDATA | 2334 | 0257 | | | | | | | | | | |
| NDISP | 1167 | 1023 | 1653 | 1667 | 2523 | 2558 | 2831 | 4255 | 5120 | | | |
| NDWRT | 1023 | 1413 | 2779 | 3753 | 4465 | 4477 | 4526 | 4727 | 5024 | | | |
| NEWK | 2167 | 2161 | | | | | | | | | | |
| NEWKI | 2181 | 2222 | | | | | | | | | | |
| NEXTI | 0509 | 3465 | 3790 | 3919 | 5096 | 5560 | | | | | | |
| NEXTA | 0691 | 0503 | 0808 | 2874 | 2908 | 3008 | 3762 | 4689 | 5404 | 5650 | | |

| | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|
| NEXTL | 0499 | 3565 | 5026 | 5348 | | | | | | | |
| NEXTS | 4910 | 4794 | | | | | | | | | |
| NGOTO | 5168 | 0278 | | | | | | | | | |
| NINIT | 3423 | 0275 | | | | | | | | | |
| NM1 | 4036 | 4027 | | | | | | | | | |
| NM2 | 4035 | 4030 | | | | | | | | | |
| NM3 | 4026 | 4039 | | | | | | | | | |
| NM4 | 4040 | 4032 | 4034 | | | | | | | | |
| NMOTA | 0200 | | | | | | | | | | |
| NMOUT | 4023 | 0200 | 3835 | 4433 | | | | | | | |
| NN | 4165 | 4155 | 4162 | 4175 | | | | | | | |
| NN1 | 4166 | 4160 | | | | | | | | | |
| NN10 | 4184 | 4182 | | | | | | | | | |
| NN11 | 4199 | | | | | | | | | | |
| NN2 | 4178 | 4130 | 4164 | | | | | | | | |
| NN3 | 4159 | 4153 | | | | | | | | | |
| NN4 | 4195 | 4185 | 4191 | 4192 | | | | | | | |
| NN5 | 4186 | 4188 | | | | | | | | | |
| NN6 | 4217 | 4199 | 4213 | | | | | | | | |
| NN7 | 4212 | 4216 | | | | | | | | | |
| NOEOF | 0857 | 0241 | 2663 | | | | | | | | |
| NOFMT | 4696 | 4710 | | | | | | | | | |
| NOPCD | 5193 | 5256 | | | | | | | | | |
| NORML | 4959 | 2286 | | | | | | | | | |
| NOST | 4867 | 0253 | | | | | | | | | |
| NOTA | 6135 | 6227 | | | | | | | | | |
| NOTOP | 6126 | | | | | | | | | | |
| NPRIN | 1168 | 1016 | 1020 | 1660 | 1666 | 2524 | | | | | |
| NPWRT | 1020 | 1375 | 2550 | 2790 | 2798 | | | | | | |
| NSTMT | 2658 | 0239 | 2649 | 2656 | | | | | | | |
| NUM | 1181 | 1182 | 6677 | 6691 | 6694 | 6704 | 6710 | | | | |
| NUMCA | 0211 | 0869 | 6209 | | | | | | | | |
| NUMOA | 0212 | 2335 | 6211 | | | | | | | | |
| NXTIL | 3593 | 3603 | 3605 | | | | | | | | |
| NXTDT | 1089 | 3656 | 5372 | 5373 | 5448 | 5646 | | | | | |
| NXTOP | 3565 | 3064 | 3628 | | | | | | | | |
| NXTP1 | 3583 | 3149 | 3573 | 3593 | 3595 | 3611 | 3618 | 3631 | | | |
| NXTP2 | 3575 | 3591 | | | | | | | | | |
| NXTP3 | 3598 | 3588 | | | | | | | | | |
| NXTP4 | 3612 | 3602 | | | | | | | | | |
| NXTPR | 3558 | 3572 | | | | | | | | | |
| NXTST | 1190 | 1386 | 3039 | 4986 | | | | | | | |
| OBLNK | 0715 | 3784 | 3811 | 3871 | 4237 | 4383 | | | | | |
| ODGIT | 1000 | 4081 | 4086 | 4186 | 4214 | | | | | | |
| OF | 0961 | 0952 | | | | | | | | | |
| OFA | 2396 | 2387 | | | | | | | | | |
| OFFA | 1629 | 1371 | | | | | | | | | |
| OFLT | 4051 | 4165 | | | | | | | | | |
| OFXD | 4114 | 4043 | | | | | | | | | |
| OL1 | 4730 | 4722 | | | | | | | | | |
| OL2 | 4731 | 4725 | | | | | | | | | |
| OL3 | 4728 | 4735 | 4738 | | | | | | | | |
| OLINE | 4719 | 0169 | 4340 | | | | | | | | |
| ONE | 0045 | 5234 | 5900 | | | | | | | | |
| ONEXT | 0716 | 1002 | 1288 | 1352 | 1400 | 1658 | 1829 | 1847 | 3748 | 3771 | 3788 |
| | 3861 | 3881 | 3884 | 3892 | 3903 | 3913 | 3915 | 4085 | 4090 | 4098 | 4103 |

CROSS-REFERENCE TABLE

| | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|
| | 4196 | 4201 | 4233 | 4750 | 5110 | | | | | | | |
| OPC1 | 6029 | 5294 | 6032 | | | | | | | | | |
| OPC2 | 6031 | 6024 | | | | | | | | | | |
| OPC3 | 6033 | 6014 | | | | | | | | | | |
| OPCHI | 6021 | 5263 | 5289 | | | | | | | | | |
| OPCHA | 0171 | 0912 | | | | | | | | | | |
| OPCHK | 6012 | 0171 | 5571 | 5972 | 5982 | | | | | | | |
| OPDMK | 0079 | 0678 | 4281 | 5483 | | | | | | | | |
| OPMSK | 0076 | 0466 | 0513 | 0546 | 2242 | 3589 | 3592 | 3819 | 3820 | 3850 | 4561 | |
| | 4560 | 4593 | 4615 | 4704 | | | | | | | | |
| OPTYF | 1182 | 6304 | 6324 | 6333 | 6465 | | | | | | | |
| OR | 2267 | 6134 | 6134 | 6135 | 6135 | | | | | | | |
| ORA | 6134 | 6281 | | | | | | | | | | |
| OROP | 6124 | | | | | | | | | | | |
| OSI | 4243 | 4231 | | | | | | | | | | |
| OSIGN | 4230 | 4080 | 4178 | | | | | | | | | |
| OTPTR | 1078 | 1706 | 1735 | 1863 | 1864 | 1873 | 1878 | 1888 | 1890 | 1899 | 1902 | |
| OUTCR | 0717 | 1350 | 1526 | 1538 | 1545 | 1950 | 1968 | | | | | |
| OUTIA | 0179 | 1301 | 2423 | 2562 | 2793 | 2800 | 3783 | 3842 | 4105 | 5108 | | |
| OVCHA | 0216 | 2171 | | | | | | | | | | |
| OVCHK | 2942 | 0216 | 2901 | 2929 | | | | | | | | |
| OVCK | 4415 | 4310 | | | | | | | | | | |
| OVFW | 0971 | 0965 | | | | | | | | | | |
| P1 | 1801 | 1812 | 1815 | 1818 | 1820 | 1826 | 1828 | 1836 | 1844 | | | |
| P2 | 1837 | 1832 | | | | | | | | | | |
| P3 | 1816 | 1810 | | | | | | | | | | |
| P4 | 1806 | 1814 | | | | | | | | | | |
| P5 | 1842 | 1822 | | | | | | | | | | |
| P6 | 1835 | 1845 | | | | | | | | | | |
| P7 | 1846 | 1802 | 1806 | | | | | | | | | |
| PAL | 1458 | 1572 | | | | | | | | | | |
| PARMA | 1091 | 5619 | 5719 | | | | | | | | | |
| PARMN | 1092 | 3659 | 5489 | 5661 | 5717 | | | | | | | |
| PBPTO | 1098 | 0561 | 0924 | 2598 | 2915 | 2924 | 2942 | 2947 | 3032 | 3184 | 3233 | |
| | 3481 | 3684 | 3706 | 6614 | | | | | | | | |
| PBPTR | 1095 | 0796 | 0925 | 1551 | 2221 | 2440 | 2444 | 2879 | 2880 | 2925 | 2927 | |
| | 2998 | 3031 | 3047 | 3062 | 3482 | 3496 | 3529 | 3626 | 3672 | 3683 | 5017 | |
| | 5104 | 5345 | 5622 | 6597 | 6613 | | | | | | | |
| PBUFA | 5696 | 5670 | | | | | | | | | | |
| PBUFF | 1096 | 0795 | 0923 | 1487 | 1496 | 1550 | 1554 | 2180 | 2181 | 2199 | 2200 | |
| | 2215 | 2218 | 2599 | 2876 | 2895 | 2996 | 3030 | 3046 | 3049 | 3127 | 3154 | |
| | 3450 | 3640 | 3655 | 3705 | 5445 | 5626 | 5644 | 5678 | 5696 | 6598 | 6612 | |
| PBUFO | 1097 | 0922 | 2216 | 2933 | 2935 | 3130 | 3483 | 3702 | | | | |
| PEND | 1150 | 0719 | 1268 | 1286 | 1290 | 1399 | 3747 | 3765 | 4362 | 4416 | 4513 | |
| | 4522 | | | | | | | | | | | |
| PEXMI | 2552 | 2530 | 2544 | | | | | | | | | |
| PEXM2 | 2525 | 2832 | | | | | | | | | | |
| PEXM3 | 2537 | 2526 | | | | | | | | | | |
| PEXMA | 0165 | 2405 | 2424 | 5074 | 5401 | | | | | | | |
| PEXMK | 2522 | 0165 | 2567 | 2569 | 2636 | 2850 | | | | | | |
| PFLAG | 1176 | 2723 | 2735 | 2737 | | | | | | | | |
| PFOR1 | 4361 | 4269 | | | | | | | | | | |
| PFORM | 4352 | 4287 | 4290 | 4406 | 4479 | | | | | | | |
| PGIN0 | 0832 | 2360 | 4946 | | | | | | | | | |
| PGINT | 0580 | 0834 | 2348 | 2355 | 6365 | 6376 | | | | | | |
| PI/2 | 0044 | | | | | | | | | | | |

CROSS-REFERENCE TABLE

| | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|
| PIA | 6050 | 6046 | | | | | | | | | |
| PLACE | 1169 | 3834 | 3864 | 3872 | 4028 | 4051 | 4128 | 4156 | 4159 | 4234 | 4429 |
| | 4505 | | | | | | | | | | |
| PLINE | 5101 | 5023 | 5116 | | | | | | | | |
| PLOTE | 4673 | | | | | | | | | | |
| PLUS | 1238 | 4093 | | | | | | | | | |
| PLUSS | 6111 | 6257 | | | | | | | | | |
| PMODE | 1114 | 1015 | 1027 | 1367 | 1370 | 1410 | 1670 | 1672 | 2507 | 2773 | 4365 |
| | 5118 | | | | | | | | | | |
| POINT | 1240 | 4084 | 4200 | | | | | | | | |
| POV | 4297 | 4312 | | | | | | | | | |
| PPFF | 1151 | | | | | | | | | | |
| PR11 | 4307 | 4304 | | | | | | | | | |
| PR17 | 4348 | 4345 | | | | | | | | | |
| PR2 | 4344 | 4336 | 4337 | 4339 | 4538 | | | | | | |
| PR3 | 4338 | 4341 | | | | | | | | | |
| PR5 | 4280 | 4271 | | | | | | | | | |
| PR6 | 4302 | 4295 | | | | | | | | | |
| PR7 | 4310 | 4279 | | | | | | | | | |
| PRADD | 1093 | 0483 | 0490 | 0497 | 0504 | 1382 | 1392 | 1395 | 2818 | 3051 | 3060 |
| | 3490 | 3506 | 3525 | 3585 | 3622 | 3625 | 3662 | 4314 | 4547 | 4552 | 4971 |
| | 4990 | 5012 | 5056 | 5094 | 5103 | 5106 | 5150 | 5152 | 5182 | 5237 | 5245 |
| | 5344 | 5362 | 5370 | 5378 | 5382 | 5398 | 5405 | 5419 | 5510 | 5651 | 5664 |
| | 6718 | 6734 | 6737 | | | | | | | | |
| PRALL | 1366 | 1458 | | | | | | | | | |
| PRCHK | 4365 | 4346 | 4420 | 4723 | | | | | | | |
| PRCHR | 4382 | 4366 | 4368 | 4395 | | | | | | | |
| PREED | 1789 | 1273 | | | | | | | | | |
| PRFLG | 1117 | 1214 | 1674 | 2806 | 4284 | 4328 | 4352 | 4360 | 4367 | 4670 | |
| PRIN1 | 2717 | 2736 | 2752 | | | | | | | | |
| PRIN2 | 2726 | 2718 | 2721 | | | | | | | | |
| PRIN3 | 2737 | 2728 | | | | | | | | | |
| PRIN4 | 2749 | 2741 | | | | | | | | | |
| PRIN5 | 2751 | 2748 | | | | | | | | | |
| PRINN | 4266 | 4338 | | | | | | | | | |
| PRINS | 2712 | 4783 | 4786 | | | | | | | | |
| PRNTS | 2722 | 2657 | 2712 | | | | | | | | |
| PROD | 1206 | 0900 | 0905 | | | | | | | | |
| PRTOP | 2684 | 2651 | | | | | | | | | |
| PTAPE | 2532 | 2282 | | | | | | | | | |
| QMRKA | 2681 | 2528 | | | | | | | | | |
| QOV | 4272 | 4299 | | | | | | | | | |
| QTCH1 | 4577 | 4242 | 4563 | 4570 | | | | | | | |
| QTCH2 | 4590 | 4575 | | | | | | | | | |
| QTCH3 | 4573 | 4565 | | | | | | | | | |
| QTCHA | 0230 | | | | | | | | | | |
| QTCHK | 4560 | 0230 | 4270 | 4475 | | | | | | | |
| QTOP | 0109 | 0161 | 0519 | 1244 | 2483 | 2683 | 3590 | 4562 | 4569 | 4621 | 4873 |
| | 6180 | | | | | | | | | | |
| QUOT | 6103 | | | | | | | | | | |
| QUOTE | 4242 | 3882 | | | | | | | | | |
| R | 1231 | 4054 | 4061 | 4064 | 4066 | 4077 | 4082 | 4087 | 4127 | 4132 | 4137 |
| | 4166 | 4197 | 4205 | 4208 | 4209 | 4212 | 4215 | | | | |
| RBASE | 1235 | 4119 | 4123 | 4139 | 4202 | | | | | | |
| RBOP | 0117 | 2686 | 2745 | 6383 | | | | | | | |
| RBRAC | 6107 | | | | | | | | | | |

PAGE 163

CROSS-REFERENCE TABLE

| | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|
| RBUFF | 1145 | 0181 | 0181 | 0198 | 0202 | 0202 | 1627 | 1627 | 2046 | 2866 | 2868 |
| | 2869 | 4769 | 4769 | 4770 | 4770 | 6641 | 6642 | | | | |
| RCAL | 2082 | 2067 | | | | | | | | | |
| RCOUT | 1118 | 4600 | 4612 | 4650 | 4673 | | | | | | |
| RDYDA | 0166 | 1390 | 1552 | 1848 | 2190 | 2202 | 2447 | 3048 | 4941 | 4948 | 4963 |
| RDYPT | 2516 | 0166 | 2980 | 3013 | | | | | | | |
| RE1 | 1740 | 1761 | | | | | | | | | |
| RE2 | 1762 | 1755 | 1758 | | | | | | | | |
| RE3 | 1743 | | | | | | | | | | |
| RE4 | 1747 | | | | | | | | | | |
| READS | 6624 | 4784 | 4787 | 6629 | | | | | | | |
| REE | 1285 | 1283 | | | | | | | | | |
| REED | 0161 | 2499 | | | | | | | | | |
| REED1 | 1267 | | | | | | | | | | |
| REED2 | 1327 | 1317 | 1320 | | | | | | | | |
| REED3 | 1336 | 1328 | 1333 | | | | | | | | |
| REED4 | 1299 | 1296 | | | | | | | | | |
| REED5 | 1344 | 1330 | 1335 | 1421 | 1451 | | | | | | |
| REED6 | 1306 | 1293 | 1299 | 1354 | 1365 | 1441 | 1508 | 1530 | 1546 | 1857 | 1862 |
| | 1866 | 1877 | 1879 | 2090 | 2110 | 2119 | 2122 | | | | |
| REED7 | 1294 | 1284 | | | | | | | | | |
| REED8 | 1417 | 1358 | | | | | | | | | |
| REED9 | 1352 | 1348 | | | | | | | | | |
| REEDD | 1356 | 1338 | | | | | | | | | |
| REEDF | 1277 | 1376 | | | | | | | | | |
| REEDG | 1308 | 1304 | | | | | | | | | |
| REEDL | 1124 | 2500 | 2566 | 3751 | | | | | | | |
| REEDT | 1414 | 1409 | 1411 | | | | | | | | |
| REFRA | 4767 | 4733 | | | | | | | | | |
| REFRS | 1735 | 1970 | 4767 | | | | | | | | |
| REMS | 6640 | 4798 | | | | | | | | | |
| RENMI | 2994 | 3012 | | | | | | | | | |
| RENM2 | 2997 | 3010 | | | | | | | | | |
| RENM3 | 3007 | | | | | | | | | | |
| RENM4 | 3011 | 2999 | 3001 | | | | | | | | |
| RENUM | 2984 | 2285 | | | | | | | | | |
| RES | 1202 | 0184 | | | | | | | | | |
| RESA | 1630 | 1452 | | | | | | | | | |
| RESB1 | 0538 | 0544 | 2082 | | | | | | | | |
| RESBP | 0544 | 2385 | 2650 | 6186 | 6231 | | | | | | |
| RET2 | 0301 | 0293 | 0810 | 0811 | 2464 | 4905 | | | | | |
| RETA | 1668 | 1667 | | | | | | | | | |
| RETN2 | 0294 | 0417 | 0444 | 0459 | 0619 | 0673 | 0781 | 0850 | 0871 | 1415 | 1728 |
| | 1841 | 1988 | 2013 | 2236 | 2468 | 2703 | 3261 | 3384 | 3397 | 3614 | 3630 |
| | 3633 | 4315 | 4321 | 4369 | 4409 | 4418 | 4424 | 4553 | 4589 | 4692 | 6340 |
| | 6403 | 6456 | 6713 | | | | | | | | |
| RETN3 | 0293 | 3269 | 3277 | 3596 | 4554 | 6315 | 6336 | | | | |
| RETN | 5707 | 5185 | | | | | | | | | |
| RETN5 | 6527 | 4792 | | | | | | | | | |
| RFNX | 2124 | 2097 | | | | | | | | | |
| RFNX1 | 1453 | 2143 | | | | | | | | | |
| RIGHT | 1899 | 1985 | | | | | | | | | |
| RIT3L | 1655 | 1647 | | | | | | | | | |
| RITE | 2707 | 4777 | | | | | | | | | |
| RITE1 | 1650 | 1664 | | | | | | | | | |
| RITE2 | 1663 | 1656 | 1661 | | | | | | | | |

PAGE 165

CROSS-REFERENCE TABLE

| | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|
| SCRAT | 2958 | 2291 | | | | | | | | | | |
| SCRK | 2967 | 2963 | | | | | | | | | | |
| SCRP | 3029 | 2512 | 2600 | | | | | | | | | |
| SEC | 1105 | | | | | | | | | | | |
| SERFI | 3516 | 3473 | | | | | | | | | | |
| SERFN | 3506 | 3451 | 3484 | 3514 | | | | | | | | |
| SERFR | 3498 | 3509 | | | | | | | | | | |
| SFLAG | 1178 | 2608 | 2700 | 6247 | 6266 | 6363 | 6369 | 6381 | 6470 | 6474 | 6578 | |
| | 6584 | | | | | | | | | | | |
| SFMT | 4661 | 4439 | | | | | | | | | | |
| SFMT1 | 4681 | 4709 | | | | | | | | | | |
| SFMT2 | 4694 | 4680 | | | | | | | | | | |
| SGNS | 1004 | 5908 | | | | | | | | | | |
| SIGN | 1184 | 0868 | 0872 | 1185 | 6208 | | | | | | | |
| SLCOD | 1113 | 1041 | 1721 | 1790 | 1838 | 2427 | 2540 | 2581 | 2787 | 4354 | 4358 | |
| | 4672 | | | | | | | | | | | |
| SLEND | 0202 | 1267 | 1529 | 3764 | | | | | | | | |
| SLWST | 0484 | 4353 | 4355 | 4448 | 4471 | 5084 | 5087 | 5090 | 5092 | 5093 | 5480 | |
| | 5557 | 5565 | 5597 | 5606 | 5608 | 5896 | 6719 | | | | | |
| SM2L | 3362 | 3352 | | | | | | | | | | |
| SMCLN | 6105 | | | | | | | | | | | |
| SNFLA | 0197 | 3832 | 5367 | 5467 | | | | | | | | |
| SPAGE | 6691 | 6674 | 6676 | | | | | | | | | |
| SPLFA | 0196 | 6031 | | | | | | | | | | |
| SPRDD | 1212 | 5396 | 5397 | 5403 | | | | | | | | |
| SSBPT | 1180 | 0533 | 0540 | | | | | | | | | |
| SSTT | 3490 | 3449 | 3661 | | | | | | | | | |
| SSYM1 | 3348 | 3355 | | | | | | | | | | |
| SSYM2 | 3376 | 3362 | 3367 | 3373 | | | | | | | | |
| SSYM3 | 3385 | 0270 | 3377 | 3380 | | | | | | | | |
| SSYM4 | 3358 | 3349 | | | | | | | | | | |
| SSYM5 | 3369 | 3364 | | | | | | | | | | |
| SSYM6 | 3353 | 3360 | | | | | | | | | | |
| SSYMA | 0170 | 5196 | 5495 | | | | | | | | | |
| SSYMT | 3328 | 0170 | 3084 | 3104 | | | | | | | | |
| ST | 1162 | 1211 | 1212 | 1213 | 4515 | 4519 | 4661 | 4708 | 5699 | | | |
| STADD | 5699 | 5702 | 5711 | | | | | | | | | |
| STAR | 2685 | 2590 | | | | | | | | | | |
| STAR2 | 0939 | | | | | | | | | | | |
| START | 0162 | 0007 | | | | | | | | | | |
| STDPG | 2686 | | | | | | | | | | | |
| STEP | 1380 | 1459 | | | | | | | | | | |
| STPA | 6098 | 6496 | | | | | | | | | | |
| STEXP | 0463 | 2352 | 2357 | 2365 | 2388 | 4919 | | | | | | |
| STMAX | 0976 | 0971 | | | | | | | | | | |
| STMPI | 3401 | 3343 | 3396 | 3691 | | | | | | | | |
| STMP2 | 3402 | 3331 | 3348 | 3361 | 3365 | 3371 | | | | | | |
| STMP3 | 3403 | 3334 | 3376 | 3381 | | | | | | | | |
| STMT | 2609 | 2640 | | | | | | | | | | |
| STMT1 | 2618 | 2614 | 2616 | | | | | | | | | |
| STOPE | 5115 | 5014 | 5016 | | | | | | | | | |
| STOPS | 2434 | 2435 | 4790 | | | | | | | | | |
| STOR1 | 0347 | 0356 | 0358 | | | | | | | | | |
| STOR2 | 0539 | 0346 | | | | | | | | | | |
| STORE | 0340 | 5227 | 5298 | | | | | | | | | |
| STOUT | 4253 | 3935 | 3979 | 4278 | | | | | | | | |

| | | | | | | | | | | | | |
|--------|------|------|------|------|------|------|------|------|------|------|------|--|
| STPI | 5123 | 5115 | 5119 | | | | | | | | | |
| STPFL | 1108 | 1380 | 1687 | 1692 | 1718 | 2538 | 2626 | 2814 | 3755 | 5010 | | |
| STPL | 1459 | 1569 | | | | | | | | | | |
| STRII | 6470 | 6463 | | | | | | | | | | |
| STRID | 6460 | 6337 | | | | | | | | | | |
| STRNP | 6472 | 0252 | | | | | | | | | | |
| STROP | 0373 | 6312 | 6334 | 6467 | 6478 | 6512 | | | | | | |
| STSRH | 0453 | 0410 | 0429 | 0467 | 0476 | 6228 | 6282 | 6492 | 6497 | 6698 | | |
| STTOP | 5982 | 5688 | 5878 | 6054 | | | | | | | | |
| SUMP | 2215 | 0227 | 2147 | 2154 | 2162 | | | | | | | |
| SUMPA | 0227 | 5643 | | | | | | | | | | |
| SVALI | 3398 | 3353 | | | | | | | | | | |
| SVALU | 3392 | 3345 | 3346 | 3399 | 3693 | | | | | | | |
| SWITH | 5371 | 5361 | | | | | | | | | | |
| SYE10 | 6498 | 0263 | | | | | | | | | | |
| SYE12 | 0874 | 0247 | | | | | | | | | | |
| SYE13 | 6625 | 0260 | | | | | | | | | | |
| SYE14 | 4872 | 0254 | | | | | | | | | | |
| SYE15 | 2713 | 0242 | | | | | | | | | | |
| SYE25 | 0618 | 0238 | 0598 | 0606 | 0612 | 0614 | 0630 | 0838 | 2534 | 3024 | | |
| SYM1 | 0398 | 0404 | | | | | | | | | | |
| SYM2 | 0407 | 0402 | | | | | | | | | | |
| SYMC1 | 0393 | 0414 | 6262 | 6371 | 6426 | 6487 | 6520 | 6656 | | | | |
| SYMC2 | 0391 | 0425 | 0438 | 2719 | 6218 | | | | | | | |
| SYMCK | 0395 | 0392 | 6256 | | | | | | | | | |
| SYME2 | 3368 | 0271 | | | | | | | | | | |
| SYMTP | 1083 | 0553 | 0564 | 2949 | 3061 | 3146 | 3241 | 3392 | 3408 | 3555 | 3557 | |
| | 5199 | 5274 | | | | | | | | | | |
| SYN13 | 4806 | 6568 | | | | | | | | | | |
| SYNE2 | 2692 | 0265 | | | | | | | | | | |
| SYNE3 | 6600 | 0267 | | | | | | | | | | |
| SYNE4 | 6454 | 0255 | 6508 | 6510 | | | | | | | | |
| SYNE5 | 6517 | 0256 | 6516 | | | | | | | | | |
| SYNE6 | 6522 | 0264 | 6489 | | | | | | | | | |
| SYNE7 | 4920 | 0258 | | | | | | | | | | |
| SYNE8 | 6486 | 0261 | 6485 | | | | | | | | | |
| SYNE9 | 6493 | 0262 | | | | | | | | | | |
| SYNTAX | 2598 | 2586 | 2589 | | | | | | | | | |
| YSER | 0524 | 0235 | | | | | | | | | | |
| SYTSA | 0209 | 3508 | | | | | | | | | | |
| SYTSH | 5343 | 0209 | 5257 | 5352 | 5380 | | | | | | | |
| TAB | 6093 | 2753 | 2753 | | | | | | | | | |
| TABA | 2753 | 2739 | | | | | | | | | | |
| TABER | 4426 | 0279 | 6060 | | | | | | | | | |
| TABOP | 4311 | 4282 | | | | | | | | | | |
| TBS2 | 2022 | 2019 | 2035 | | | | | | | | | |
| TBS4 | 2034 | 2031 | | | | | | | | | | |
| TBS5 | 2020 | 2010 | | | | | | | | | | |
| TBS6 | 2008 | 2025 | | | | | | | | | | |
| TBSR1 | 2014 | 1337 | 1357 | | | | | | | | | |
| TBSRH | 2015 | 0168 | 1420 | | | | | | | | | |
| TEMP | 1135 | 1136 | 2618 | 2632 | 2865 | 2882 | 2886 | 2893 | 2900 | 6192 | 6672 | |
| | 6695 | 6701 | 6709 | | | | | | | | | |
| TEMP1 | 1137 | 0847 | 1061 | 4874 | 4890 | 4896 | 5198 | 5275 | 5330 | 5636 | 5642 | |
| | 6179 | 6190 | 6252 | 6310 | 6311 | 6332 | 6368 | 6380 | 6447 | 6455 | 6464 | |
| | 6475 | | | | | | | | | | | |

PAGE 167

CROSS-REFERENCE TABLE

| | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|
| TEMP2 | 1138 | 0821 | 0849 | 2943 | 4889 | 4894 | 5213 | 5264 | 5295 | 5306 | 5309 |
| | 5343 | 5350 | 5674 | 5999 | 6314 | 6327 | 6335 | 6468 | 6513 | 6582 | 6588 |
| TEMP3 | 1139 | 0794 | 0802 | 0812 | 0822 | 1404 | 2169 | 2877 | 2891 | 2892 | 2910 |
| | 2976 | 2978 | 4699 | 4707 | 5205 | 5680 | 5994 | | | | |
| TEMP4 | 1140 | 0584 | 0594 | 0613 | 0627 | 0637 | 0638 | 0824 | 0827 | 1402 | 2194 |
| | 2417 | 2911 | 2918 | 2919 | 2921 | 2946 | 4960 | 5219 | 5221 | 5671 | 6003 |
| TEMPS | 1134 | 0498 | 0499 | 0502 | 0505 | 0506 | 0508 | 0512 | 0677 | 1224 | 1225 |
| | 1226 | 1227 | 1229 | 2605 | 2997 | 3000 | 3006 | 3007 | 3092 | 3093 | 3094 |
| | 3094 | 3100 | 3101 | 3461 | 3462 | 3470 | 3474 | 3492 | 3510 | 3516 | 3523 |
| | 3569 | 3570 | 3575 | 3579 | 3580 | 3583 | 3584 | 3612 | 3613 | 3648 | 3678 |
| | 3734 | 3740 | 3766 | 3774 | 3775 | 3778 | 3785 | 3792 | 3793 | 3794 | 3814 |
| | 3818 | 3838 | 3839 | 3844 | 3845 | 3849 | 3927 | 3932 | 3937 | 3942 | 4280 |
| | 4313 | 4316 | 4405 | 4443 | 4546 | 4560 | 4566 | 4567 | 4571 | 4577 | 4580 |
| | 4581 | 4587 | 4662 | 4674 | 4675 | 4677 | 4679 | 4691 | 4698 | 4972 | 5079 |
| | 5086 | 5163 | 5164 | 5171 | 5180 | 5236 | 5328 | 5347 | 5364 | 5368 | 5371 |
| | 5374 | 5394 | 5406 | 5418 | 5429 | 5444 | 5468 | 5482 | 5486 | 5509 | 5512 |
| | 5550 | 5558 | 5610 | 5612 | 5624 | 5628 | 5640 | 5649 | 5656 | 5662 | 5666 |
| | 5684 | 6151 | 6202 | 6239 | 6270 | 6390 | 6396 | 6397 | 6400 | 6401 | 6423 |
| | 6424 | 6435 | 6436 | 6439 | 6731 | 6732 | | | | | |
| TEPPP | 1211 | 5407 | 5421 | 5428 | | | | | | | |
| TERM | 1149 | 1034 | 1676 | 1684 | 2521 | 2785 | 2815 | 4291 | 4294 | 4331 | 4334 |
| | 4344 | 4423 | 5391 | | | | | | | | |
| TEST | 5327 | 5259 | 5276 | | | | | | | | |
| TFLAG | 1103 | 1271 | 1693 | 2517 | 2532 | 2775 | 2803 | | | | |
| TFLGI | 1107 | 4953 | 4959 | 4962 | 5002 | 5006 | 5019 | 5012 | | | |
| THENA | 4923 | 4918 | | | | | | | | | |
| TIMES | 6113 | 6657 | | | | | | | | | |
| TMP | 1215 | 1440 | 1517 | 1518 | 1521 | 1523 | 1524 | 1532 | 1534 | 1539 | 1649 |
| | 1736 | 2042 | 2099 | 3962 | 3978 | 4747 | 4751 | | | | |
| TMP1 | 1216 | 1231 | 1648 | 1663 | 1740 | 1750 | 1756 | 1796 | 1811 | 3961 | 3970 |
| TMP10 | 1225 | 3763 | 3773 | 3846 | 3929 | 3936 | | | | | |
| TMP11 | 1226 | 3735 | 3809 | 3812 | 3869 | | | | | | |
| TMP12 | 1227 | 3817 | 3887 | 3893 | 3901 | | | | | | |
| TMP13 | 1228 | 3796 | 3806 | 4716 | 4184 | 4187 | | | | | |
| TMP2 | 1217 | 1232 | 1738 | 1813 | 1816 | 2016 | 2020 | 2536 | 4297 | 4309 | 4380 |
| | 4583 | 4588 | | | | | | | | | |
| TMP2A | 2042 | 1697 | 1745 | | | | | | | | |
| TMP3 | 1218 | 1234 | 1744 | 1749 | 2022 | 2024 | 3964 | 3973 | 3975 | 3976 | |
| TMP4 | 1219 | 1235 | 1936 | 1959 | 1963 | 2015 | 2032 | 2034 | | | |
| TMP5 | 1220 | 1233 | 1303 | 1318 | 1323 | 1325 | 1331 | 1341 | 1344 | 1359 | 1363 |
| | 1794 | 1839 | 1921 | 1924 | 1942 | 1948 | 1964 | 1983 | 2014 | 2173 | 2235 |
| TMP6 | 1221 | 1321 | 1329 | 1449 | 1512 | 1981 | | | | | |
| TMP6A | 1222 | 2018 | 2026 | 4748 | 4749 | | | | | | |
| TMP7 | 1223 | 1503 | 3729 | 3741 | 3772 | | | | | | |
| TMP9 | 1224 | 3791 | 3875 | | | | | | | | |
| TMPA | 0203 | 0762 | 1525 | 1537 | 1544 | 1741 | | | | | |
| TMPOP | 0184 | 4300 | 4431 | 6048 | | | | | | | |
| TOA | 6133 | 6491 | | | | | | | | | |
| TOBLN | 1632 | 1710 | | | | | | | | | |
| TRAC2 | 4952 | 4955 | | | | | | | | | |
| TRAC3 | 4961 | 4954 | | | | | | | | | |
| TRACE | 4955 | 2287 | | | | | | | | | |
| TREGE | 1157 | | | | | | | | | | |
| TRUE | 5900 | 5925 | 5931 | 5938 | | | | | | | |
| TSRET | 6699 | 6693 | | | | | | | | | |
| TSRHA | 0168 | 0453 | 0463 | 2066 | 2740 | | | | | | |

| | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|
| TSTPT | 1085 | 5062 | 5091 | 5130 | 5472 | 5605 | 5620 | 5687 | 5848 | 5854 | 5879 |
| | 5884 | 5887 | 5944 | 5988 | 5990 | 6013 | 6033 | 6035 | | | |
| TTY | 1039 | 1030 | 1662 | 4502 | | | | | | | |
| TYPE | 1210 | 3073 | 3086 | 3114 | 3568 | | | | | | |
| TYPEC | 4600 | 4444 | | | | | | | | | |
| TYPFL | 0078 | 2743 | 3815 | | | | | | | | |
| UI | 1493 | 1491 | | | | | | | | | |
| U3 | 1498 | 1477 | 1494 | | | | | | | | |
| U4 | 1487 | 1474 | 1490 | | | | | | | | |
| U5 | 1501 | 1478 | 1479 | 1488 | 1497 | | | | | | |
| UFLAG | 1177 | 6153 | 6196 | 6216 | 6225 | 6251 | 6402 | 6405 | | | |
| UKDEF | 2665 | 2591 | | | | | | | | | |
| UKERR | 2667 | 0243 | | | | | | | | | |
| UKPTR | 1094 | 0920 | 0927 | 2217 | 2546 | 2665 | 2928 | 2932 | 3128 | 3129 | 3479 |
| | 3703 | 6601 | | | | | | | | | |
| UNEQL | 6118 | | | | | | | | | | |
| UNFW | 0967 | | | | | | | | | | |
| UNITS | 1104 | 2506 | 3045 | 4928 | | | | | | | |
| UNMIN | 6120 | 6219 | | | | | | | | | |
| UNSTK | 0491 | 4357 | 4359 | 4481 | 4483 | 5060 | 5061 | 5063 | 5065 | 6736 | |
| UPARA | 0191 | 5873 | | | | | | | | | |
| UPARW | 1489 | 1565 | | | | | | | | | |
| UPLUS | 6121 | | | | | | | | | | |
| VALUE | 1082 | 0554 | 3166 | 3174 | 3231 | 3391 | 3425 | 3435 | 3439 | 3442 | 3692 |
| VALUP | 1599 | 1336 | | | | | | | | | |
| VAR01 | 6316 | 6308 | | | | | | | | | |
| VAR03 | 6335 | | | | | | | | | | |
| VAR04 | 6337 | 6306 | 6329 | 6331 | | | | | | | |
| VAR06 | 6328 | 6321 | | | | | | | | | |
| VAROA | 0214 | 4910 | 6579 | 6624 | | | | | | | |
| VAROP | 6301 | 0214 | 6155 | 6484 | 6515 | | | | | | |
| WAITS | 6502 | 4789 | 6523 | 6530 | | | | | | | |
| WBUFF | 0309 | 4525 | | | | | | | | | |
| WIDTH | 1170 | 1230 | 3867 | 3876 | 4071 | 4074 | 4174 | 4495 | | | |
| WRIT | 1647 | 0159 | | | | | | | | | |
| WRIT2 | 4444 | 4455 | 4466 | 4532 | | | | | | | |
| WRIT4 | 4472 | 4478 | | | | | | | | | |
| WRIT5 | 4479 | 4476 | | | | | | | | | |
| WRIT6 | 4529 | 4503 | | | | | | | | | |
| WRIT7 | 4505 | 4496 | | | | | | | | | |
| WRIT8 | 4536 | 4474 | 4531 | | | | | | | | |
| WRITB | 4490 | 4485 | | | | | | | | | |
| WRITE | 0159 | 1018 | 1021 | 1024 | | | | | | | |
| WRITF | 4470 | 4449 | | | | | | | | | |
| WRITN | 4456 | 4445 | | | | | | | | | |
| WRITQ | 4465 | 4446 | | | | | | | | | |
| WRITS | 4454 | 4456 | | | | | | | | | |
| WRITX | 4461 | 4451 | | | | | | | | | |
| XC | 1197 | 1215 | 1216 | 1217 | 1218 | 3491 | 3493 | 3495 | 3497 | 3522 | 3524 |
| | 3526 | 3528 | | | | | | | | | |
| XEC | 4981 | | | | | | | | | | |
| XEC3 | 4986 | | | | | | | | | | |
| XEC4 | 4995 | 0164 | 5038 | 5151 | 5159 | 5319 | 5438 | 5450 | 5705 | 6543 | 6549 |
| | 6559 | 6560 | 6561 | 6562 | | | | | | | |
| XEC4A | 0164 | 4348 | 4760 | 4765 | 4929 | 4973 | 6741 | | | | |
| XEC5 | 5014 | 5011 | | | | | | | | | |

CROSS-REFERENCE TABLE

[illegible]

PAGE 170

9830 MATH ROUTINES

```

0001          ASMB,A,L,C
0003* MATH ROUTINES 3-20-72
0004*
0005 00000          ORG 0
0006*
0007 00000 164121    JMP START,I  START OF MONITOR
0008 00001 043060    OCT 43060    CONSTANT NEEDED BY PROCESSOR
0009 00002 000000    BSS 1
0010*
0011 00003 000000    C6          OCT 0          ONE
0012 00004 010000    OCT 010000    0000
0013 00005 000000    OCT 0          0000
0014*
0015 00006 000000    C5          OCT 0000000    PI/2
0016 00007 012560    OCT 12560    1570
0017 00010 074543    OCT 074543    7963
0018 00011 023200    OCT 023200    2680
0019*
0020 00012 000001    .1          DEC 1
0021*
0022 00013 000400    C7          OCT 000400    180/PI
0023 00014 053451    OCT 053451    5729
0024 00015 053571    OCT 053571    5779
0025 00016 050450    OCT 050450    5128
0026*
0027 00017 000002    .2          DEC 2
0028 00020 000003    .3          DEC 3
0029 00021 000004    .4          DEC 4
0030 00022 000005    .5          DEC 5
0031 00023 000006    .6          DEC 6
0032 00024 000007    .7          DEC 7
0033 00025 000010    .8          DEC 8
0034 00026 000011    .9          DEC 9
0035 00027 000013    .11         DEC 11
0036 00030 000014    .12         DEC 12
0037 00031 000015    CRTN       OCT 15
0038 00031          EOL          EQU CRTN
0039 00031          .13         EQU CRTN
0040*
0041* POINTERS TO FUNCTIONS CONSTANTS
0042*
0043 00027          DPERR       EQU .11    POINTER TO 180/PI
0044 00023          PI/2        EQU .6     POINTER TO PI/2
0045 00020          ONE         EQU .3     POINTER TO THE CONSTANT ONE
0046*
0047*
0048 00032 000017    .15         DEC 15
0049 00032 000020    .16         DEC 16
0050 00034 000025    .21         DEC 21
0051 00035 000026    .22         DEC 22
0052 00036 000027    .23         DEC 23
0053 00037 000034    .28         DEC 28
0054 00040 000037    .31         DEC 31
0055 00041 000040    .32         DEC 32
0056 00042 000041    .33         DEC 33
0057 00043 000042    .34         DEC 34

```

PAGE 171

9830 MATH ROUTINES

| | | | | | |
|------|-------|--------|-------|-----|--------|
| 0058 | 00044 | 000045 | .37 | DEC | 37 |
| 0059 | 00045 | 000050 | .40 | DEC | 40 |
| 0060 | 00046 | 000051 | .41 | DEC | 41 |
| 0061 | 00047 | 000053 | .43 | DEC | 43 |
| 0062 | 00050 | 000055 | .45 | DEC | 45 |
| 0063 | 00051 | 000056 | .46 | DEC | 46 |
| 0064 | 00052 | 000060 | .48 | DEC | 48 |
| 0065 | 00053 | 000072 | .58 | DEC | 58 |
| 0066 | 00054 | 000077 | .63 | DEC | 63 |
| 0067 | 00055 | 000111 | I | OCT | 111 |
| 0068 | 00056 | 023420 | .1E4 | DEC | 10000 |
| 0069 | 00057 | 001750 | .1000 | DEC | 1000 |
| 0070 | 00060 | 000144 | .100 | DEC | 100 |
| 0071 | 00061 | 000012 | .10 | DEC | 10 |
| 0072 | 00061 | | LNFD | EQU | .10 |
| 0073 | 00062 | 100000 | FLGBT | OCT | 100000 |
| 0074 | 00063 | 000143 | .99 | DEC | 99 |
| 0075 | 00064 | 100004 | INTFL | OCT | 100004 |
| 0076 | 00065 | 076000 | OPMSK | OCT | 76000 |
| 0077 | 00066 | 077777 | INF | OCT | 77777 |
| 0078 | 00067 | 100037 | TYPFL | OCT | 100037 |
| 0079 | 00070 | 101777 | OPDMK | OCT | 101777 |
| 0080 | 00071 | 000105 | E | OCT | 105 |
| 0081 | 00072 | 000106 | F | OCT | 106 |
| 0082 | 00073 | 000123 | S | OCT | 123 |
| 0083 | 00032 | | B17 | EQU | .15 |
| 0084 | 00033 | | B20 | EQU | .16 |
| 0085 | 00074 | 000022 | B22 | OCT | 22 |
| 0086 | 00075 | 000023 | B23 | OCT | 23 |
| 0087 | 00035 | | B26 | EQU | .22 |
| 0088 | 00036 | | B27 | EQU | .23 |
| 0089 | 00076 | 000036 | B36 | OCT | 36 |
| 0090 | 00040 | | B37 | EQU | .31 |
| 0091 | 00043 | | B42 | EQU | .34 |
| 0092 | 00077 | 000044 | B44 | OCT | 44 |
| 0093 | 00100 | 000052 | B52 | OCT | 52 |
| 0094 | 00053 | | B72 | EQU | .58 |
| 0095 | 00101 | 000100 | B100 | OCT | 100 |
| 0096 | 00102 | 000133 | B133 | OCT | 133 |
| 0097 | 00103 | 000140 | B140 | OCT | 140 |
| 0098 | 00104 | 000177 | B177 | OCT | 177 |
| 0099 | 00105 | 000377 | B377 | OCT | 377 |
| 0100 | 00013 | | B400 | EQU | C7 |
| 0101 | 00106 | 000777 | B777 | OCT | 777 |
| 0102 | 00107 | 001000 | B1000 | OCT | 1000 |
| 0103 | 00110 | 001400 | B1400 | OCT | 1400 |
| 0104 | 00111 | 001740 | B1740 | OCT | 1740 |
| 0105 | 00112 | 001760 | B1760 | OCT | 1760 |
| 0106 | 00113 | 001774 | B1774 | OCT | 1774 |
| 0107 | 00114 | 001775 | B1775 | OCT | 1775 |
| 0108 | 00115 | 001777 | B1777 | OCT | 1777 |
| 0109 | 00116 | 002000 | QTOP | OCT | 2000 |
| 0110 | 00117 | 002400 | B2400 | OCT | 2400 |
| 0111 | 00120 | 003000 | B3000 | OCT | 3000 |
| 0112 | 00121 | 004000 | B4000 | OCT | 4000 |
| 0113 | 00122 | 006000 | B6K | OCT | 6000 |

PAGE 172

9830 MATH ROUTINES

| | | | | | |
|------|-------|--------|-------|-----|--------|
| 0114 | 00004 | | B10K | EQU | C6+1 |
| 0115 | 00004 | | RPOP | EQU | B10K |
| 0116 | 00123 | 012000 | B12K | OCT | 12000 |
| 0117 | 00123 | | RBOP | EQU | B12K |
| 0118 | 00124 | 032000 | EXPOP | OCT | 32000 |
| 0119 | 00125 | 050000 | LBOP | OCT | 50000 |
| 0120 | 00126 | 052000 | LPOP | OCT | 52000 |
| 0121 | 00127 | 170000 | B170K | OCT | 170000 |
| 0122 | 00130 | 013777 | FROMA | OCT | 13777 |
| 0123 | 00131 | 000000 | LROMA | BSS | 1 |
| 0124 | 00132 | 177777 | M1 | DEC | -1 |
| 0125 | 00133 | 177776 | M2 | DEC | -2 |
| 0126 | 00134 | 177775 | M3 | DEC | -3 |
| 0127 | 00135 | 177774 | M4 | DEC | -4 |
| 0128 | 00136 | 177773 | M5 | DEC | -5 |
| 0129 | 00137 | 177772 | M6 | DEC | -6 |
| 0130 | 00140 | 177767 | M9 | DEC | -9 |
| 0131 | 00141 | 177766 | M10 | DEC | -10 |
| 0132 | 00142 | 177765 | M11 | DEC | -11 |
| 0133 | 00143 | 177764 | M12 | DEC | -12 |
| 0134 | 00144 | 177763 | M13 | DEC | -13 |
| 0135 | 00145 | 177762 | M14 | DEC | -14 |
| 0136 | 00146 | 177761 | M15 | DEC | -15 |
| 0137 | 00147 | 177751 | M23 | DEC | -23 |
| 0138 | 00150 | 177747 | M25 | DEC | -25 |
| 0139 | 00151 | 177740 | M32 | DEC | -32 |
| 0140 | 00152 | 177734 | M36 | DEC | -36 |
| 0141 | 00153 | 177720 | M48 | DEC | -48 |
| 0142 | 00154 | 177700 | M64 | DEC | -64 |
| 0143 | 00155 | 177660 | M80 | DEC | -80 |
| 0144 | 00156 | 177634 | M100 | DEC | -100 |
| 0145 | 00157 | 177400 | M256 | DEC | -256 |
| 0146 | 00160 | 176000 | M1024 | DEC | -1024 |
| 0147 | 00161 | 154360 | MAXSN | DEC | -10000 |

-(MAXIMUM ALLOWABLE LINE
NUMBER)

| | | | | | |
|-------|---|--------|-------|-----|--------|
| 0148 | 00162 | 043116 | FN | ASC | 1, FN |
| 0149* | | | | | |
| 0150* | MATH CONSTANTS THAT ARE NEEDED ON BASE PAGE | | | | |
| 0151* | | | | | |
| 0152 | 00163 | 177401 | ZAP | OCT | 177401 |
| 0153 | 00164 | 063440 | MASK1 | OCT | 063440 |
| 0154 | 00165 | 061400 | MASK2 | OCT | 061400 |

PAGE 173

9830 MATH ROUTINES

| | | | | | | | |
|-------|-------|--------|-------|-----|-------|---------------------------------|--------|
| 0156* | | | | | | | |
| 0157 | 00166 | 000000 | FWAM | BSS | 1 | | |
| 0158 | 00112 | | RSTKA | EQU | B1760 | START OF RETURN STACK | |
| 0159 | 00167 | 000000 | WRITE | BSS | 1 | PRINTER AND DISPLAY DRIVER | |
| 0160 | 00170 | 000000 | CLERA | BSS | 1 | CLEAR SCREEN | |
| 0161 | 00116 | | REED | EQU | QTOP | KEYBOARD DRIVER QTOP IS = 2000B | |
| 0162 | 00121 | | START | EQU | B4000 | | |
| 0163 | 00171 | 000000 | RUNA | BSS | 1 | PHASE Z:SYMBOL TABLE AND | |
| | | | | | | STORAGE | |
| 0164 | 00172 | 000000 | XEC4A | BSS | 1 | | |
| 0165 | 00173 | 000000 | PEXMA | BSS | 1 | | |
| 0166 | 00174 | 000000 | RDYDA | BSS | 1 | | |
| 0167 | 00122 | | LISTA | EQU | B6K | TABLE SEARCH | |
| 0168 | 00175 | 000000 | TSRHA | BSS | 1 | OUTPUT I/O BUFFER | |
| 0169 | 00176 | 000000 | ELINA | BSS | 1 | SEARCH SYMBOL TABLE FOR SYMBOL | |
| 0170 | 00177 | 000000 | SSYMA | BSS | 1 | | |
| 0171 | 00200 | 000000 | OPCHA | BSS | 1 | | |
| 0172 | 00201 | 000000 | FORMA | BSS | 1 | | |
| 0173 | 00202 | 000000 | FSCA | BSS | 1 | | |
| 0174 | 00203 | 000000 | ACTSA | BSS | 1 | | |
| 0175 | 00204 | 000000 | DELSA | BSS | 1 | | |
| 0176 | 00205 | 000000 | AERRA | BSS | 1 | ALTERNATIVE ENTRY FOR ERROR | |
| 0177 | 00206 | 000000 | ERRA | BSS | 1 | | |
| 0178 | 00207 | 000000 | INTEA | BSS | 1 | | |
| 0179 | 00210 | 017036 | OUTIA | DEF | OUTIN | OUTPUT AN INTEGER | |
| 0180 | 00211 | 000000 | .BUFA | BSS | 1 | FIRST CHAR OF I/O BUFFER | |
| 0181 | 00212 | 000000 | .SLBF | BSS | 1 | REFRESH BUFFER POINTER | |
| 0182 | 00213 | 001744 | ARIA | DEF | AR1 | | |
| 0183 | 00214 | 001754 | AR2A | DEF | AR2 | | |
| 0184 | 00215 | 000000 | TMPOP | BSS | 1 | | |
| 0185 | 00216 | 011500 | .FADA | DEF | FAD | | |
| 0186 | 00217 | 011466 | .FSBA | DEF | FSB | SUBTRACT | |
| 0187 | 00220 | 016311 | .FMPA | DEF | FMP | MULTIPLY | |
| 0188 | 00221 | 011576 | .FDVA | DEF | FDV | DIVIDE | |
| 0189 | 00222 | 000000 | ENTFA | BSS | 1 | | |
| 0190 | 00223 | 016162 | .FSR | DEF | FSR | | |
| 0191 | 00224 | 016054 | UPARA | DEF | UPARO | UP ARROW | |
| 0192 | 00225 | 017021 | FLTRA | DEF | FLTRF | | |
| 0193 | 00226 | 017001 | FLTFA | DEF | FLTFX | | |
| 0194 | 00227 | 017055 | FXFLA | DEF | FXFLT | | |
| 0195 | 00230 | 017121 | FLTSA | DEF | FLTSP | | |
| 0196 | 00231 | 016365 | SPLFA | DEF | SPLFL | | |
| 0197 | 00232 | 017331 | SNFLA | DEF | SNFLT | SYNTAX FORMAT TO FLOATING | |
| 0198 | 00233 | 000000 | BEND | BSS | 1 | I/O BUFFER END | |
| 0199 | 00233 | | SBUFA | EQU | BEND | | |
| 0200 | 00234 | 000000 | NMOTA | BSS | 1 | NUMBER OUTPUT ROUTINE LINK | |
| 0201 | 00235 | 000000 | FILLA | BSS | 1 | SUPPLY NEEDED BLANKS ON | OUTPUT |
| 0202 | 00236 | 000000 | SLEND | BSS | 1 | REFRESH BUFFER END | |
| 0203 | 00111 | | TMPA | EQU | B1740 | | |
| 0204 | 00110 | | BADRP | EQU | B1400 | POINTER POINTER TO I/O BUFFER | |
| 0205* | | | | | | | |
| 0206 | 00123 | | FOPBS | EQU | B12K | | |
| 0207 | 00237 | 000000 | SALTA | BSS | 1 | | |
| 0208 | 00240 | 000000 | MDIMA | BSS | 1 | | |
| 0209 | 00241 | 000000 | SYTSA | BSS | 1 | | |
| 0210 | 00242 | 000000 | DCOMA | BSS | 1 | | |
| 0211 | 00243 | 017154 | NUMCA | DEF | NUMCK | | |

PAGE 174

9830 MATH ROUTINES

| | | | | | |
|------|-------|--------|-------|-----|-------|
| 0212 | 00244 | 017266 | NUMOA | DEF | NUMOP |
| 0213 | 00245 | 000000 | GTSTA | BSS | 1 |
| 0214 | 00246 | 000000 | VAROA | BSS | 1 |
| 0215 | 00247 | 000000 | ECALA | BSS | 1 |
| 0216 | 00250 | 000000 | OVCHA | BSS | 1 |
| 0217 | 00251 | 000000 | CLPIA | BSS | 1 |
| 0218 | 00252 | 000000 | ENTEA | BSS | 1 |
| 0219 | 00253 | 000000 | CHRSA | BSS | 1 |
| 0220 | 00254 | 000000 | XSCRA | BSS | 1 |
| 0221 | 00255 | 000000 | XEC6A | BSS | 1 |
| 0222 | 00256 | 000000 | FDAAA | BSS | 1 |
| 0223 | 00257 | 000000 | FNDAA | BSS | 1 |
| 0224 | 00260 | 000000 | FNDIA | BSS | 1 |
| 0225 | 00261 | 000000 | IOSTA | BSS | 1 |
| 0226 | 00262 | 000000 | INREA | BSS | 1 |
| 0227 | 00263 | 000000 | SUMPA | BSS | 1 |
| 0228 | 00264 | 000000 | CRCKA | BSS | 1 |
| 0229 | 00265 | 000000 | IOHEA | BSS | 1 |
| 0230 | 00266 | 000000 | QTCHA | BSS | 1 |

0231 *****

0232*

| | | | | | |
|------|-------|--------|-----|---|-----------------------------------|
| 0233 | 00267 | 000000 | BSS | 1 | CHECKSUM WORD FOR FIRST 512 BLOCK |
|------|-------|--------|-----|---|-----------------------------------|

PAGE 175

9830—ERROR TABLE

| | | | | | |
|------|-------|--------|-----|-----|---------|
| 0235 | 00270 | 000000 | E1 | BSS | |
| 0236 | 00271 | 000000 | | BSS | |
| 0237 | 00272 | 000000 | | BSS | |
| 0238 | 00273 | 000000 | | BSS | |
| 0239 | 00274 | 000000 | E5 | BSS | |
| 0240 | 00275 | 000000 | | BSS | |
| 0241 | 00276 | 000000 | | BSS | |
| 0242 | 00277 | 000000 | | BSS | |
| 0243 | 00300 | 000000 | | BSS | |
| 0244 | 00301 | 000000 | E10 | BSS | |
| 0245 | 00302 | 017255 | | DEF | NUMER |
| 0246 | 00303 | 017172 | | DEF | NUMCP+1 |
| 0247 | 00304 | 000000 | | BSS | |
| 0248 | 00305 | 000000 | | BSS | |
| 0249 | 00306 | 000000 | E15 | BSS | |
| 0250 | 00307 | 000000 | | BSS | |
| 0251 | 00310 | 000000 | | BSS | |
| 0252 | 00311 | 000000 | | BSS | |
| 0253 | 00312 | 000000 | | BSS | |
| 0254 | 00313 | 000000 | E20 | BSS | |
| 0255 | 00314 | 000000 | | BSS | |
| 0256 | 00315 | 000000 | | BSS | |
| 0257 | 00316 | 000000 | | BSS | |
| 0258 | 00317 | 000000 | | BSS | |
| 0259 | 00320 | 000000 | E25 | BSS | |
| 0260 | 00321 | 000000 | | BSS | |
| 0261 | 00322 | 000000 | | BSS | |
| 0262 | 00323 | 000000 | | BSS | |
| 0263 | 00324 | 000000 | | BSS | |
| 0264 | 00325 | 000000 | E30 | BSS | |
| 0265 | 00326 | 000000 | | BSS | |
| 0266 | 00327 | 000000 | | BSS | |
| 0267 | 00330 | 000000 | | BSS | |
| 0268 | 00331 | 000000 | | BSS | |
| 0269 | 00332 | 000000 | E35 | BSS | |
| 0270 | 00333 | 000000 | | BSS | |
| 0271 | 00334 | 000000 | | BSS | |
| 0272 | 00335 | 000000 | | BSS | |
| 0273 | 00336 | 000000 | | BSS | |
| 0274 | 00337 | 000000 | E40 | BSS | |
| 0275 | 00340 | 000000 | | BSS | |
| 0276 | 00341 | 000000 | | BSS | |
| 0277 | 00342 | 000000 | | BSS | |
| 0278 | 00343 | 000000 | | BSS | |
| 0279 | 00344 | 000000 | E45 | BSS | |
| 0280 | 00345 | 000000 | | BSS | |
| 0281 | 00346 | 000000 | | BSS | |
| 0282 | 00347 | 000000 | | BSS | |
| 0283 | 00350 | 000000 | | BSS | |
| 0284 | 00351 | 000000 | E50 | BSS | |
| 0285 | 00352 | 013355 | | DEF | ER47 |
| 0286 | 00353 | 016207 | | DEF | ER48 |
| 0287 | 00354 | 016053 | | DEF | ER49 |
| 0288 | 00355 | 016141 | | DEF | ER50 |

LOG OF A NEGATIVE NUMBER
 SQUARE ROOT OF A NEGATIVE NUMBER
 ZERO TO THE ZERO POWER
 NEGATIVE NUMBER RAISED TO A
 NON-INTEGERS

PAGE 176

9830—BASE PAGE READ-WRITE MEMORY

| | | | | |
|-------|-------|--------|-------------|-----------|
| 0290 | 01400 | | | ORG 1400B |
| 0291 | 01400 | 000000 | BADDR | BSS 1 |
| 0292 | 01401 | 000000 | RSARE | BSS 1 |
| 0293 | 01402 | 000000 | OTPTR | BSS 1 |
| 0294 | 01403 | 000000 | MAW | BSS 1 |
| 0295 | 01404 | 000000 | LWAM | BSS 1 |
| 0296 | 01405 | 000000 | FWUP | BSS 1 |
| 0297 | 01406 | 000000 | VALUE | BSS 1 |
| 0298 | 01407 | 000000 | SYMTP | BSS 1 |
| 0299 | 01410 | 000000 | LSTAK | BSS 1 |
| 0300 | 01411 | 000000 | TSTPT | BSS 1 |
| 0301 | 01412 | 000000 | LSTPT | BSS 1 |
| 0302 | 01413 | 000000 | HSTPT | BSS 1 |
| 0303 | 01414 | 000000 | DSTRT | BSS 1 |
| 0304 | 01415 | 000000 | NXTDT | BSS 1 |
| 0305 | 01416 | 000000 | MODE | BSS 1 |
| 0306 | 01417 | 000000 | PARMA | BSS 1 |
| 0307 | 01420 | 000000 | PARMN | BSS 1 |
| 0308 | 01421 | 000000 | PRADD | BSS 1 |
| 0309 | 01422 | 000000 | UKPTR | BSS 1 |
| 0310 | 01423 | 000000 | PBPTR | BSS 1 |
| 0311 | 01424 | 000000 | PBUFF | BSS 1 |
| 0312 | 01425 | 000000 | PBUFO | BSS 1 |
| 0313 | 01426 | 000000 | PBPTO | BSS 1 |
| 0314 | 01427 | 000000 | LNUM | BSS 1 |
| 0315 | 01430 | 000000 | LNINC | BSS 1 |
| 0316 | 01431 | 000000 | CLINE | BSS 1 |
| 0317 | 01432 | 000000 | DRQST | BSS 1 |
| 0318 | 01433 | 000000 | TFLAG | BSS 1 |
| 0319 | 01434 | 000000 | UNITS | BSS 1 |
| 0320 | 01435 | 000000 | SEC | BSS 1 |
| 0321 | 01436 | 000000 | HFLGI | BSS 2 |
| 0322 | 01440 | 000000 | TFLGI | BSS 2 |
| 0323 | 01442 | 000000 | STPFL | BSS 1 |
| 0324 | 01443 | 000000 | KEYFG | BSS 1 |
| 0325 | 01444 | 000000 | SAVEA | BSS 1 |
| 0326 | 01445 | 000000 | SAVEB | BSS 1 |
| 0327 | 01446 | 000000 | ADRSS | BSS 1 |
| 0328 | 01447 | 000000 | SLCOD | BSS 1 |
| 0329 | 01450 | 000000 | PMODE | BSS 1 |
| 0330 | 01451 | 000000 | CTYPE | BSS 1 |
| 0331 | 01452 | 000000 | CPLAC | BSS 1 |
| 0332 | 01453 | 000000 | PRFLG | BSS 1 |
| 0333 | 01454 | 000000 | RCOUT | BSS 1 |
| 0334 | 01455 | 000000 | FORMT | BSS 1 |
| 0335 | 01456 | 000000 | FORMS | BSS 1 |
| 0336 | 01457 | 000000 | FORME | BSS 1 |
| 0337 | 01460 | 000000 | FTMP1 | BSS 1 |
| 0338 | 01461 | 000000 | FTMP2 | BSS 1 |
| 0339 | 01462 | 000000 | REEDL | BSS 1 |
| 0340 | 01463 | 000000 | INTEL | BSS 1 |
| 0341 | 01464 | 000000 | | BSS 2 |
| 0342* | | | | |
| 0343* | | | TEMPORARIES | |
| 0344* | | | | |
| 0345 | 01466 | 000000 | LOCL1 | BSS 1 |

CURRENT CHAR POINTER
 SAVE (A) FOR RETN2
 REFRESH POINTER
 LAST WORD OF R/W MEM
 LAST WORD AVAILABLE FOR USER
 FIRST WORD OF COMMON OR
 MAINLINE
 FIRST WORD -1 OF VALUE TABLE
 FIRST WORD -1 OF SYMBOL TABLE
 FIRST WORD OF OPERATOR STACK
 TEMPORARY STACK POINTER
 OPERATOR STACK POINTER
 OPERAND STACK POINTER
 DATA STATEMENT
 POINTERS
 PROGRAM=0: KEYBOARD=1
 PARAMETER ADDRESS
 PARAMETER NAME
 LAST WORD+1 OF CURRENT STMT
 USER KEY POINTER
 LAST WORD OF CURRENT PROGRAM
 FIRST WORD OF CURRENT PROGRAM
 FIRST WORD OF MAINLINE
 LAST WORD+1 OF MAINLINE
 CURRENT LINE NUMBER
 AUTO LINE INCREMENT
 CURRENT LINE FOR LIST
 INPUT STMT FLAG
 PTAPE FLAG
 TRIG UNITS
 SECURE PROGRAM FLAG
 STOP ADDRESSES
 TRACE LIMIT ADDRESSES
 STOP FLAG
 KEYBOARD FLAG
 INTERRUPT
 TEMPORARIES
 SELECT CODE
 PRINT-ALL FLAG
 KEYBOARD
 FORMAT FLAGS
 PRINT/DISP FLAG
 FORMAT
 STATEMENT FLAGS
 CURRENT OPERATOR CODE
 CURRENT OPERATOR PRECEDENCE
 UNUSED

PAGE 177

9830 - - BASE PAGE READ-WRITE MEMORY

| | | | | | | |
|-------|--|--------|-------|-----|------|---|
| 0346 | 01467 | 000000 | LOCL2 | BSS | 1 | |
| 0347 | 01470 | 000000 | L2T1 | BSS | 1 | |
| 0348 | 01471 | 000000 | L2T2 | BSS | 1 | |
| 0349 | 01472 | 000000 | TEMPS | BSS | 1 | |
| 0350 | 01473 | 000000 | TEMP | BSS | 1 | |
| 0351 | 01473 | | MBOX1 | EQU | TEMP | |
| 0352 | 01474 | 000000 | TEMP1 | BSS | 1 | |
| 0353 | 01475 | 000000 | TEMP2 | BSS | 1 | |
| 0354 | 01476 | 000000 | TEMP3 | BSS | 1 | |
| 0355 | 01477 | 000000 | TEMP4 | BSS | 1 | |
| 0356* | | | | | | |
| 0357* | BUFFERS | | | | | |
| 0358* | | | | | | |
| 0359 | 01500 | 000000 | IOBUF | BSS | 41 | |
| 0360 | 01551 | 000000 | RBUF | BSS | 41 | INPUT AND RECALL BUFFER
SINGLE LINE REFRESH BUFFER |
| 0361* | | | | | | |
| 0362* | PRINT ROUTINE FLAGS | | | | | |
| 0363* | | | | | | |
| 0364 | 01622 | 000000 | TERM | BSS | 1 | |
| 0365 | 01623 | 000000 | PEND | RSS | 1 | |
| 0366 | 01624 | 000000 | PPFF | BSS | 1 | |
| 0367 | 01625 | 000000 | FIRST | BSS | 1 | |
| 0368 | 01626 | 000000 | | BSS | 2 | CRLF FLAG
BUFFER OVERFLOW FLAG
BUFFER POINTER OR TAB FLAG |
| 0369* | | | | | | UNUSED |
| 0370* | MATH TEMPORARIES | | | | | |
| 0371* | | | | | | |
| 0372 | 01630 | 000000 | TREGE | BSS | 4 | |
| 0373 | 01634 | 000000 | RSLTE | BSS | 4 | |
| 0374* | | | | | | |
| 0375* | STATEMENT SYNTAX AND EXECUTION TEMPORARIES | | | | | |
| 0376* | | | | | | |
| 0377 | 01640 | 000000 | ST | BSS | 32 | |
| 0378* | | | | | | |
| 0379* | SYSTEM SYNTAX AND EXECUTION TEMPORARIES | | | | | |
| 0380* | | | | | | |
| 0381 | 01700 | 000000 | | BSS | 1 | |
| 0382 | 01701 | 000000 | NDISP | BSS | 1 | |
| 0383 | 01702 | 000000 | NPRIN | BSS | 1 | |
| 0384 | 01703 | 000000 | PLACE | BSS | 1 | |
| 0385 | 01704 | 000000 | WIDTH | BSS | 1 | |
| 0386 | 01705 | 000000 | FLOAT | BSS | 1 | |
| 0387 | 01706 | 000000 | SBPTR | BSS | 1 | |
| 0388 | 01707 | 000000 | BLANK | BSS | 1 | |
| 0389 | 01710 | 000000 | MSELG | BSS | 1 | |
| 0390 | 01711 | 000000 | DFLAG | BSS | 1 | |
| 0391 | 01712 | 000000 | PFLAG | BSS | 1 | |
| 0392 | 01713 | 000000 | UFLAG | BSS | 1 | |
| 0393 | 01714 | 000000 | SFLAG | BSS | 1 | |
| 0394 | 01715 | 000000 | SBADR | BSS | 1 | |
| 0395 | 01716 | 000000 | SSBPT | BSS | 1 | |
| 0396 | 01717 | 000000 | NUM | BSS | 1 | |
| 0397 | 01717 | | OPTYM | EQU | NUM | |
| 0398 | 01720 | 000000 | LASTN | BSS | 1 | |
| 0399 | 01721 | 000000 | SIGN | BSS | 1 | |
| 0400 | 01721 | | MAXIN | EQU | SIGN | |
| 0401 | 01722 | 000000 | EXP | BSS | 1 | |

SAVED FOR PROCESSOR

PAGE 178 9830 - - BASE PAGE READ-WRITE MEMORY

| | | | | | |
|------|-------|---------|-------|-----|-----|
| 0402 | 01722 | | INTGR | EQU | EXP |
| 0403 | 01723 | 0000000 | ARYAD | BSS | 1 |
| 0404 | 01724 | 0000000 | GFLAG | BSS | 1 |
| 0405 | 01725 | 0000000 | NXTST | BSS | 1 |
| 0406 | 01726 | 0000000 | MBINI | BSS | 1 |
| 0407 | 01727 | 0000000 | MBIN2 | BSS | 1 |

0408*

ARITHMETIC REGISTERS AND RETURN STACK

0409*

0410*

| | | | | | |
|------|-------|---------|-------|-----|-------|
| 0411 | 01740 | | | ORG | 1740B |
| 0412 | 01740 | 0000000 | XC | BSS | 4 |
| 0413 | 01744 | 0000000 | ARI | BSS | 4 |
| 0414 | 01750 | 0000000 | YC | BSS | 4 |
| 0415 | 01754 | 0000000 | AR2 | BSS | 4 |
| 0416 | 01760 | 0000000 | RSTAK | BSS | 11 |
| 0417 | 01773 | 0000000 | RES | BSS | 4 |
| 0418 | 01777 | 0000000 | RSPTR | BSS | 1 |

0419*

RETURN STACK
RESULT REGISTOR
RETURN STACK POINTER

294

PAGE 179

NON-COMMON BASE PAGE ROM

| | | | | |
|-------|-------|-------|-----|-------|
| 0421 | 01744 | ARIE | EQU | ARI |
| 0422 | 01754 | AR2E | EQU | AR2 |
| 0423 | 00136 | N5 | EQU | M5 |
| 0424 | 01237 | STAR2 | EQU | 1237B |
| 0425 | 01305 | ROUND | EQU | 1305B |
| 0426 | 01261 | OF | EQU | 1261B |
| 0427 | 01274 | STMAX | EQU | 1274B |
| 0428* | | | | |
| 0429 | 00356 | RETN3 | EQU | 356B |
| 0430 | 00357 | RETN2 | EQU | 357B |
| 0431 | 00365 | RET2 | EQU | 365B |
| 0432 | 00373 | ICNT | EQU | 373B |
| 0433 | 00374 | WBUFF | EQU | 374B |
| 0434 | 00400 | GPARM | EQU | 400B |
| 0435 | 00405 | XFAR1 | EQU | 405B |
| 0436 | 00412 | RSULT | EQU | 412B |
| 0437 | 00416 | STORE | EQU | 416B |
| 0438 | 00425 | STOR1 | EQU | 425B |
| 0439 | 00444 | XFAR2 | EQU | 444B |
| 0440 | 00450 | STROP | EQU | 450B |
| 0441 | 00456 | SBPUD | EQU | 456B |
| 0442 | 00461 | SYMC2 | EQU | 461B |
| 0443 | 00463 | SYMC1 | EQU | 463B |
| 0444 | 00464 | SYMCK | EQU | 464B |
| 0445 | 00504 | COMCK | EQU | 504B |
| 0446 | 00510 | COMCE | EQU | 510B |
| 0447 | 00513 | LPCK | EQU | 513B |
| 0448 | 00521 | LPCKE | EQU | 521B |
| 0449 | 00524 | RPCK | EQU | 524B |
| 0450 | 00534 | RPCKE | EQU | 534B |
| 0451 | 00535 | FSCE2 | EQU | 535B |
| 0452 | 00537 | STSRH | EQU | 537B |
| 0453 | 00546 | STEXP | EQU | 546B |
| 0454 | 00553 | BHSTP | EQU | 553B |
| 0455 | 00561 | MER8 | EQU | 561B |
| 0456 | 00561 | FSCE4 | EQU | 561B |
| 0457 | 00563 | SLWST | EQU | 563B |
| 0458 | 00567 | UNSTK | EQU | 567B |
| 0459 | 00574 | NEXTL | EQU | 574B |
| 0460 | 00606 | NEXT1 | EQU | 606B |
| 0461 | 00613 | FPAGE | EQU | 613B |
| 0462 | 00626 | SAVBP | EQU | 626B |
| 0463 | 00633 | RESB1 | EQU | 633B |
| 0464 | 00640 | RESBP | EQU | 640B |
| 0465 | 00645 | MINIT | EQU | 645B |
| 0466 | 00651 | INITS | EQU | 651B |
| 0467 | 00664 | LNUM | EQU | 664B |
| 0468 | 00666 | PGINT | EQU | 666B |
| 0469 | 00676 | INTCK | EQU | 676B |
| 0470 | 00726 | SYE25 | EQU | 726B |
| 0471 | 00730 | GETAD | EQU | 730B |
| 0472 | 00742 | FND | EQU | 742B |
| 0473 | 00750 | DIGCK | EQU | 750B |
| 0474 | 00752 | LETCK | EQU | 752B |
| 0475 | 00753 | LIMIT | EQU | 753B |
| 0476 | 00771 | GTOPP | EQU | 771B |

PAGE 180

NON-COMMON BASE PAGE ROM

| | | | | |
|------|-------|-------|-----|-------|
| 0477 | 01007 | NEXTA | EQU | 1007B |
| 0478 | 01015 | OBLNK | EQU | 1015B |
| 0479 | 01016 | ONEXT | EQU | 1016B |
| 0480 | 01017 | OUTCR | EQU | 1017B |
| 0481 | 01041 | GETCH | EQU | 1041B |
| 0482 | 01053 | GTMP | EQU | 1053B |
| 0483 | 01055 | GNEXT | EQU | 1055B |
| 0484 | 01062 | GFRST | EQU | 1062B |
| 0485 | 01066 | GETSK | EQU | 1066B |
| 0486 | 01075 | FNDPS | EQU | 1075B |
| 0487 | 01121 | MVTOH | EQU | 1121B |
| 0488 | 01131 | PGIN0 | EQU | 1131B |
| 0489 | 01140 | LTR | EQU | 1140B |
| 0490 | 01147 | ENDS | EQU | 1147B |
| 0491 | 01151 | EOST | EQU | 1151B |
| 0492 | 01152 | NOEOF | EQU | 1152B |
| 0493 | 01153 | EOLCK | EQU | 1153B |
| 0494 | 01156 | CONST | EQU | 1156B |
| 0495 | 01165 | SYE12 | EQU | 1165B |
| 0496 | 01166 | GSIGN | EQU | 1166B |
| 0497 | 01201 | IMPY | EQU | 1201B |
| 0498 | 01217 | FETCH | EQU | 1217B |
| 0499 | 01225 | LDEF | EQU | 1225B |
| 0500 | 01254 | ZONK | EQU | 1254B |
| 0501 | 01256 | CLAR2 | EQU | 1256B |
| 0502 | 01314 | ODGIT | EQU | 1314B |
| 0503 | 01317 | SGNS | EQU | 1317B |
| 0504 | 01330 | .NPWR | EQU | 1330B |
| 0505 | 01335 | NPWRT | EQU | 1335B |
| 0506 | 01337 | NDWRT | EQU | 1337B |
| 0507 | 01342 | CRLF | EQU | 1342B |
| 0508 | 01344 | CRLF1 | EQU | 1344B |
| 0509 | 01351 | TTY | EQU | 1351B |
| 0510 | 01362 | GTCON | EQU | 1362B |
| 0511 | 01371 | FIX | EQU | 1371B |

296

284

1,444,141

284

PAGE 181

CONSTANTS AND EQUATES FOR BOTH

```

0513                                     LST
0514*      MIKE WINGERT
0515*      AND GENE BURMEISTER 8-24-71
0516*
0517*
0518*****
0519*
0520*      BASE PAGE CONSTANTS
0521*
0522      00012      P1      EQU .1
0523      00017      P2      EQU .2
0524      00021      P4      EQU .4
0525      00022      P5      EQU .5
0526      00025      P8      EQU .8
0527      00030      P12     EQU .12
0528      00031      P13     EQU CRTN
0529      00032      P15     EQU .15
0530      00013      P256    EQU B400
0531      00132      N1      EQU M1
0532      00141      N10     EQU M10
0533      00157      N256    EQU M256
0534      00057      ATEMP   EQU .1000      POINTER TO YC REGISTER
0535      00004      D1000   EQU B10K
0536*
0537*****
0538*
0539*      SHARED READ/WRITE
0540*
0541      01701      TEMPA    EQU 1701B
0542      01466      TEMP5    EQU LOCL1
0543*
0544*
0545*****
0546*
0547*      R/W MEMORY AREA
0548*
0549      01702      ADR1     EQU 1702B
0550      01703      ADR2     EQU 1703B
0551      01703      ADR3     EQU ADR2
0552      01704      TEMP     EQU 1704B      TEMPORARY
0553      01705      TABL0    EQU TEMP+1    CORDIC CONSTANT
0554      01706      GFG      EQU TEMP+2    G FLAG
0555      01707      SHIFT    EQU TEMP+3    SHIFT FACTOR
0556      01710      COUNT    EQU TEMP+4    BCD DATA STORAGE
0557      01711      RCPFG    EQU TEMP+5    RECIPROCAL FLAG
0558      01712      FNCFG    EQU TEMP+6    FUNCTION FLAG
0559      01713      RY1      EQU TEMP+7    REGISTER ADDRESS STORAGE
0560      01714      SAVE     EQU TEMP+8    TEMPORARY
0561      01715      RY2      EQU TEMP+9    REGISTER ADDRESS STORAGE
0562      01716      SIGN     EQU TEMP+10   EXPONENT WORD STORAGE
0563      01717      DIGIT    EQU TEMP+11   DIGIT STACK POINTER
0564      01720      WRDN     EQU TEMP+12   START OF DIGIT STACK
0565      01734      ADRYE    EQU 1734B
0566*
0567*
0568*****

```

297

PAGE 182

CONSTANTS AND EQUATES FOR BOTH

0569*

0570***

LABELS

0571*

0572 00023

P6

EQU .6

+6

0573 00024

P7

EQU .7

+7

0574 00061

P10

EQU .10

+10

0575 00133

N2

EQU M2

-2

0576 00135

N4

EQU M4

-4

0577 00143

N12

EQU M12

-12

0578 00062

D8000

EQU FLGBT

BCD 8000

0579*

0580*

LINKAGE TO ARITHMETIC

0581*

0582 00216

ADD

EQU .FADA

FLOATING ADD ROUTINE

0583 00220

MLT

EQU .FMPA

FLOATING MULTIPLY ROUTINE

0584 00221

DVD

EQU .FDVA

FLOATING DIVIDE ROUTINE

0585*

0586*

0587*****

0588*

PAGE 183

LOG CONSTANTS AT 11600B

0591 03700 ORG 3700B
 0592*
 0593*
 0594*****
 0595*
 0596* ROM AREA
 0597*
 0598 03700 000000 C1 OCT 000000 LN(2)
 0599 03701 003223 OCT 003223 0693
 0600 03702 012161 OCT 012161 1471
 0601 03703 100126 OCT 100126 8056
 0602*
 0603 03704 000000 DEC 0000 LN(1.1)
 0604 03705 004523 OCT 004523 0953
 0605 03706 010027 OCT 010027 1017
 0606 03707 114004 OCT 114004 9804
 0607*
 0608 03710 000000 DEC 0000 LN(1.01)
 0609 03711 004625 OCT 004625 0995
 0610 03712 001460 OCT 001460 0330
 0611 03713 102462 OCT 102462 8532
 0612*
 0613 03714 000000 DEC 0000 LN(1.001)
 0614 03715 004631 OCT 004631 0999
 0615 03716 050003 OCT 050003 5003
 0616 03717 031410 OCT 031410 3308
 0617*
 0618 03720 000000 DEC 0000 LN(1.0001)
 0619 03721 004631 OCT 004631 0999
 0620 03722 112400 OCT 112400 9500
 0621 03723 001463 OCT 001463 0333
 0622*
 0623 03724 000000 OCT 000000 LN(1.00001)
 0624 03725 004631 OCT 004631 0999
 0625 03726 114520 OCT 114520 9950
 0626 03727 000003 OCT 000003 0003
 0627*
 0628 03730 000000 C2 OCT 000000 LN(10)
 0629 03731 021402 OCT 021402 2302
 0630 03732 054120 OCT 054120 5850
 0631 03733 111400 OCT 111400 9300
 0632*
 0633***
 0634*
 0635 03734 000000 DEC 0000 ARCTAN(1)
 0636 03735 003605 OCT 003605 0785
 0637 03736 034601 OCT 034601 3981
 0638 03737 061500 OCT 061500 6340
 0639*
 0640 03740 000000 DEC 0000 ARCTAN(.1)
 0641 03741 004626 OCT 004626 0996
 0642 03742 064145 OCT 064145 6865
 0643 03743 022221 OCT 022221 2491
 0644*
 0645 03744 000000 DEC 0000 ARCTAN(.01)
 0646 03745 004631 OCT 004631 09999

PAGE 184

LOG CONSTANTS AT 11600B

| | | | | | |
|-------|-------|--------|-----|--------|----------------|
| 0647 | 03746 | 113146 | OCT | 113146 | 9666 |
| 0648 | 03747 | 064147 | OCT | 064147 | 6867 |
| 0649* | | | | | |
| 0650 | 03750 | 000000 | DEC | 0000 | ARCTAN(.001) |
| 0651 | 03751 | 004631 | OCT | 004631 | 0999 |
| 0652 | 03752 | 114626 | OCT | 114626 | 9996 |
| 0653 | 03753 | 063147 | OCT | 063147 | 6667 |
| 0654* | | | | | |
| 0655 | 03754 | 000000 | OCT | 000000 | ARCTAN(.0001) |
| 0656 | 03755 | 004631 | OCT | 004631 | 0999 |
| 0657 | 03756 | 114631 | OCT | 114631 | 9999 |
| 0658 | 03757 | 113147 | OCT | 113147 | 9667 |
| 0659* | | | | | |
| 0660 | 03760 | 000000 | OCT | 000000 | ARCTAN(.00001) |
| 0661 | 03761 | 004631 | OCT | 004631 | 0999 |
| 0662 | 03762 | 114631 | OCT | 114631 | 9999 |
| 0663 | 03763 | 114627 | OCT | 114627 | 9997 |
| 0664* | | | | | |
| 0665 | 03764 | 000000 | C3 | OCT | 0000000 |
| 0666 | 03765 | 030501 | OCT | 030501 | PI |
| 0667 | 03766 | 054446 | OCT | 054446 | 3141 |
| 0668 | 03767 | 051540 | OCT | 051540 | 5926 |
| 0669* | | | | | 5360 |
| 0670 | 03770 | 000000 | C4 | OCT | 0000000 |
| 0671 | 03771 | 061203 | OCT | 061203 | 2PI |
| 0672 | 03772 | 014123 | OCT | 014123 | 6283 |
| 0673 | 03773 | 003440 | OCT | 003440 | 1853 |
| 0674* | | | | | 0720 |
| 0675 | 03774 | 000400 | C8 | OCT | 000400 |
| 0676 | 03775 | 061546 | OCT | 061546 | 200/PI |
| 0677 | 03776 | 014567 | OCT | 014567 | 6366 |
| 0678 | 03777 | 021545 | OCT | 021545 | 1977 |
| | | | | | 2365 |

300

PAGE 185

FUNCTIONS FOR 9830

| | | | | | |
|-------|-------------------------------|--------|-------|------------|----------------------------------|
| 0681 | 12533 | | ORG | 12533B | |
| 0682* | | | | | |
| 0683 | 12533 | 003774 | GPERR | DEF C8 | ADDRESS OF 200/PI |
| 0684 | 00035 | | P22 | EQU .22 | |
| 0685 | 00060 | | P100 | EQU .100 | |
| 0686 | 00144 | | N13 | EQU M13 | |
| 0687 | 00156 | | N100 | EQU M100 | |
| 0688 | 12534 | 003730 | LN10 | DEF C2 | ADDRESS OF LN10 |
| 0689 | 12535 | 003764 | PI | DEF C3 | ADDRESS OF PI |
| 0690 | 12536 | 003770 | TWOPI | DEF C4 | ADDRESS OF 2PI |
| 0691 | 12537 | 001720 | AWRDN | DEF WRDN | TOP OF THE DIGIT STACK |
| 0692 | 00111 | | ADRX | EQU B1740 | ADDRESS OF REGISTER X |
| 0693 | 12540 | 001734 | ADRY | OCT 1734 | ADDRESS OF REGISTER Y |
| 0694 | 12540 | | ADR0 | EQU ADRY | ADDRESS OF REGISTOR THETE |
| 0695 | 12541 | 001630 | TREG | DEF TREGE | ADDRESS OF REGISTOR T |
| 0696 | 12542 | 001634 | RSLT | DEF RSLTE | ADDRESS OF THE ORIGINAL ARGUMENT |
| 0697 | 00110 | | EP3 | EQU B1400 | EXPONENT POSITIVE 3 |
| 0698 | 00013 | | D0100 | EQU B400 | BCD 0100 |
| 0699 | 00033 | | D0010 | EQU B20 | BCD 0010 |
| 0700 | 00121 | | GRAF | EQU B4000 | EXPONENT POSITIVE 8 |
| 0701 | 12543 | 176400 | EN3 | OCT 176400 | EXPONENT NEGATIVE 3 |
| 0702 | 12544 | 172000 | EN12 | OCT 172000 | EXPONENT NEGATIVE 12 |
| 0703 | 12545 | 003674 | THETL | DEF C1-4 | ADDRESS OF LOG CONSTANT TABLE |
| 0704 | 12546 | 003724 | THETT | DEF C2-4 | ADDRESS OF TRIG CONSTANT TABLE |
| 0705 | 00012 | | RAD | EQU P1 | RADIAN CODE |
| 0706 | 00017 | | GRA | EQU P2 | GRAD CODE |
| 0707* | | | | | |
| 0708* | ADDITIONAL CONSTANTS FOR MATH | | | | |
| 0709* | | | | | |
| 0710 | 00117 | AD5 | EQU | B2400 | |
| 0711 | 00145 | N14 | EQU | M14 | |

PAGE 186

FUNCTIONS FOR 9830

0713*
0714*
0715*****
0716*
0717*** PROGRAM SUBROUTINES
0718*
0719*
0720*****
0721*
0722* INITIALIZE ROUTINE
0723*
0724 12547 035712 INIT STB FNCFG STORE THE FUNCTION FLAG
0725*
0726 12550 031703 STA ADR2 STORE THE ADDRESS OF THE
ARGUMENT
0727 12551 121703 LDA ADR2,I SAVE THE ORIGINAL EXPONENT
0728 12552 031716 STA SIGN WORD IN SIGN
0729*
0730 12553 021703 LDA ADR2 SAVE THE ANSWER OF THE RESULT
0731 12554 026542 LDB RSLT
0732 12555 035703 STB ADR2
0733 12556 170004 XFR SO IT IS SURELY IN A R/W
TEMPORARY
0734*
0735 12557 022542 RTT LDA RSLT LOAD THE ARGUMENT
0736 12560 024214 LDB AR2A INTO THE AR2 WORKING REGISTER
0737 12561 170004 XFR
0738*
0739 12562 066635 JMP ZERO JUMP TO DETERMINE IF IT IS ZERO
0740*
0741*
0742*****
0743*
0744* NORMALIZE TWO ARGUMENTS
0745*
0746 12563 035702 DIV STB ADRI SAVE THE ADDRESS IN B
0747 12564 062575 JSM NORM NORMALIZE THE NO. SPECIFIED IN
A
0748 12565 026541 LDB TREG TRANSFER IT INTO
0749 12566 170004 XFR A TEMPORARY
0750 12567 021702 LDA ADRI LOAD THE OTHER ADDRESS
0751 12570 062575 JSM NORM NORMALIZE IT
0752 12571 026542 ITR LDB RSLT TRANSFER IT INTO THE
0753 12572 020214 LDA AR2A RESULT REGISTER AND RET
0754 12573 170004 XFR
0755 12574 170402 RET
0756*
0757*
0758*****
0759*
0760* NORMALIZE A SINGLE ARGUMENT
0761*
0762 12575 062560 NORM JSM RTT+1 TRANSFER THE ARGUMENT
SPECIFIED IN A IN
0763 12576 171450 NRM NORMALIZE IT
0764 12577 074404 SBL 8 ALIGN THE SHIFT COUNT IN B
0765 12600 074076 TCB COMPLEMENT IT
0766 12601 005754 ADB AR2E ADD THE EXPONENT WORD TO IT
0767 12602 035754 STB AR2E RESTORE THE EXPONENT WORD
0768 12603 020214 LDA AR2A LOAD THE ADD. OF THE ARGUMENT

302

PAGE 187

FUNCTIONS FOR 9830

| | | | | |
|-------|---|--------|---------------|---|
| 0769 | 12604 | 170402 | RET | AND RETURN |
| 0770* | | | | |
| 0771* | | | | |
| 0772* | ***** | | | |
| 0773* | | | | |
| 0774* | REDUCE ONE NUMBER TO A FRACTIONAL PART OF ANOTHER | | | |
| 0775* | | | | |
| 0776 | 12605 | 062727 | FRAC JSM FDV1 | DIVIDE THE ARGUMENT BY THE SPECIFIED CONSTANT |
| 0777* | | | | |
| 0778 | 12606 | 025754 | LDB AR2E | LOAD THE EXPONENT WORD OF THE RESULT |
| 0779 | 12607 | 070742 | SAR 16 | CLEAR THE COUNT WORD |
| 0780 | 12610 | 031710 | STA COUNT | FOR STORING THE INTEGER PART |
| 0781 | 12611 | 074252 | SBM *+5 | SKIP IF THE EXPONENT IS NEGATIVE |
| 0782* | | | | |
| 0783 | 12612 | 004157 | ADB N256 | SUBTRACT 1 FROM THE EXPONENT |
| 0784 | 12613 | 171400 | MLS | LEFT SHIFT THE ARGUMENT |
| 0785 | 12614 | 001710 | ADA COUNT | STORE THE LEFT |
| 0786 | 12615 | 070604 | SAL 4 | SHIFTED DIGIT |
| 0787 | 12616 | 031710 | STA COUNT | IN COUNT |
| 0788 | 12617 | 075553 | SBP *-5 | SKIP IF THE EXPONENT IS STILL POSITIVE |
| 0789 | 12620 | 035754 | STB AR2E | STORE THE NEW EXPONENT WORD |
| 0790* | | | | |
| 0791 | 12621 | 062576 | JSM NORM+1 | JUMP TO NORMALIZE THE ARGUMENT |
| 0792* | | | | |
| 0793 | 12622 | 021702 | LDA ADR1 | MULTIPLY THIS FRACTIONAL |
| 0794 | 12623 | 066750 | JMP FMP-1 | PART OF THE ORIGINAL SPECIFIED CONSTANT |
| 0795* | | | | |
| 0796* | | | | |
| 0797* | ***** | | | |
| 0798* | | | | |
| 0799* | UNNORMALIZE AN ARGUMENT | | | |
| 0800* | | | | |
| 0801 | 12624 | 025754 | UNRM LDB AR2E | LOAD THE EXPONENT WORD |
| 0802 | 12625 | 074413 | SBP ZERO | SKIP IF THE EXPONENT IS POSITIVE |
| 0803 | 12626 | 074340 | ABR 8 | ALIGN AND COMPLEMENT |
| 0804 | 12627 | 074076 | TCB | THE EXPONENT |
| 0805 | 12630 | 074117 | LDA B | DUPLICATE IT AND |
| 0806 | 12631 | 000143 | ADA N12 | CHECK TO SEE IF IT |
| 0807 | 12632 | 070112 | SAM *+2 | IS GREATER THAN 12 |
| 0808 | 12633 | 024031 | LDB P13 | SET B=13 IF IT IS |
| 0809* | | | | |
| 0810 | 12634 | 061305 | JSM | ROUND SHIFT AND COUNT THE ARGUMENT |



PAGE 188

FUNCTIONS FOR 9830

0812*

0813*

0814*****

0815*

0816* DETERMINE WHETHER AN ARGUMENT IS ZERO

0817*

0818 12635 070075 ZERO SEC *+1,C CLEAR THE E REGISTER

0819 12636 021755 LDA AR2E+1 ADD THE 12 DIGITS

0820 12637 001756 ADA AR2E+2 OF AR2 TOGETHER

0821 12640 001757 ADA AR2E+3 IN A

0822 12641 072154 SES *+3,S SKIP IF THE REGISTER OVERFLOWED
AND SET E

0823 12642 072110

0824 12643 070075

0825 12644 025754

0826 12645 170402

0827*

0828*

0829*****

0830*

0831* RECIPROCATE AN ARGUMENT

0832*

0833 12646 062571 RECIP JSM ITR

0834 12647 020020 LDA ONE SET THE DIVIDEND=1

0835 12650 025703 LDB ADR3

0836 12651 066731 JMP .FDV+1

304

PAGE 189

FUNCTIONS FOR 9830

```

0893*
0894*
0895*****
0896*
0897* INITIALIZE ROUTINE FOR THE TRIG FUNCTIONS
0898*
0899*
0900 12714 074742 STN SBR 16 CLEAR THE 6 FLAG
0901 12715 035706 STB GFG AND THE TEMP
0902 12716 035704 STB TEMP LOCATION
0903 12717 062547 JSM INIT FETCH THE ARGUMENT AND
INITIALIZE OTHER FLAGS
0904*
0905 12720 025434 LDB UNITS LOAD THE UNITS WORD
0906 12721 074250 SZB DTOR IF IT IS SET TO DEGREES JUMP TO
CONVERT TO RAD.
0907 12722 014012 CPB RAD IF IT SAYS NOT RADIANS CONVERT
0908 12723 170402 RET IT FROM GRADS TO RADIANS
0909*
0910 12724 026533 GTOR LDB GPERR LOAD THE ADDRESS OF 200/PI
0911 12725 066727 JMP DTOR+1 DIVIDE IT INTO THE ARGUMENT
0912*
0913*
0914*****
0915*
0916* LINK ROUTINE TO FLOATING DIVIDE
0917*
0918 12726 024027 DTOR LDB DPERR LOAD THE ADDRESS OF 180/PI
0919 12727 021703 FDV1 LDA ADR3
0920 12730 031703 .FDV STA ADR2
0921 12731 160221 JSM DVD,I DO THE DIVIDE
0922 12732 066737 JMP SANS STORE THE RESULT
0923*
0924*
0925*****
0926*
0927* LINK ROUTINE TO FLOATING ADD
0928*
0929 12733 022542 FADI LDA RSLT
0930 12734 035702 .FAD STB ADR1
0931 12735 031703 STA ADR2
0932 12736 160216 JSM ADD,I DO THE ADD
0933 12737 020214 SANS LDA AR2A
0934 12740 025703 LDB ADR3
0935 12741 170004 XFR
0936 12742 170402 RET

```

PAGE 190

FUNCTIONS FOR 9830

0938*

0939*

0940*****

0941*

0942* LINK ROUTINE TO FLOATING MULTIPLY

0943*

0944 12743 020027 RTOD LDA DPERR

0945 12744 026541 LDB TREG

0946 12745 170004 XFR

0947 12746 045633 ISZ TREG+3

0948 12747 022541 LDA TREG

0949 12750 024214 LDB AR2A

0950*

0951*

0952 12751 160220 FMP JSM MLT,1

0953 12752 066737 JMP SANS

0954*

0955*

0956*****

0957*

0958* REGISTER TRANSFER SUBROUTINES

0959*

0960 12753 020020 OTO LDA ONE

0961 12754 024213 XARI LDB ARIA

0962 12755 170004 XFR

0963 12756 170402 RET

0964*

0965 12757 020020 OTT LDA ONE

0966 12760 066560 JMP RTT+1

LOAD THE ADDRESS OF 180/PI
TRANSFER THIS QUANTITY INTO A
TEMPORARY IN ORDER TO
INCREMENT DIGIT 12 BY 1
SET (A) TO THE POINTER TO TREG
SET B=ADD. OF AR2

STORE THE ANSWER

SET A=THE ADDRESS OF FLOATING
ONE
TRANSFER IT INTO AR1 AND RETURN

SET A=THE ADDRESS OF FLOATING
ONE
TRANSFER FLOATING ONE INTO AR2

PAGE 191

FUNCTIONS FOR 9830

0968*****

0969*

0970*** MAIN PROGRAMS

0971*

0972*****

0973*

0974* FIND THE SINE OF AN ARGUMENT

0975*

0976 12761 062714 SIN JSM STN

FETCH THE ARGUMENT AND
INITIALIZE FLAGS

0977 12762 045704 ISZ TEMP

SET TEMP=1

0978 12763 063054 JSM TAN+1

CALCULATE THE TANGENT OF THE
ARGUMENT

0979*

0980 12764 072150 RZA *+3

SKIP IF OVERFLOW DID NOT OCCUR
IF TAN WAS INFINITY

0981 12765 062757 JSM OTT

SET + OR - ONE INTO AR2

0982 12766 066773 JMP .CS

0983 12767 062737 JSM SANS

0984 12770 070742 SAR 16

LET THE FUNCTION FLAG BE 0
JUMP TO CALCULATE SQR(1+XX)

0985 12771 062652 JSM SCRT

0986*

0987 12772 062730 JSM .FDV

DIVIDE THE ORIGINAL NO. BY THIS
QUANTITY

0988*

0989 12773 021754 .CS LDA AR2E

LOAD THE EXPONENT WORD WORD
OF THE RESULT

0990 12774 072051 SLA *+1,S

SET THE MANTISSA SIGN BIT
LOAD THE RECIPROCAL FLAG INTO

0991 12775 025711 LDB RCPFG

B

0992 12776 004133 ADB N2

SUBTRACT 2 FROM IT
LET BIT 0=1 IF THE RESULT WAS

0993 12777 074702 SBR 15

NEGATIVE
ADD THE ORIGINAL MANTISSA

0994 13000 005716 ADB SIGN

SIGN BIT TO THIS
IF THE RESULTANT LOW BIT IS

0995 13001 074111 SLB *+2

ZERO CLEAR THE MANTISSA SIGN
BIT OF THE ANSWER

0996 13002 070071 SLA *+1,C

SET B=0 AND JUMP

0997 13003 074742 SBR 16

0998 13004 070112 SAM *+2

0999 13005 035757 STB AR2+3

1000 13006 067202 JMP OUT

TO THE STORE ROUTINE

1001*

1002*****

1003*

1004* FIND THE COSINE OF AN ARGUMENT

1005*

1006 13007 062714 COS JSM STN

FETCH THE ARGUMENT IN
RADIANS

1007 13010 021754 LDA AR2E

CHECK THE EXPONENT

1008 13011 070340 AAR 8

IF LESS THAN 11

1009 13012 000142 ADA M11

SKIP

1010 13013 070112 SAM *+2

OTHERWISE ASSUME A FULL
CIRCLE

1011 13014 066757 JMP OTT

1012 13015 024023 LDB PI/2

ADD PI/2 TO THE ARGUMENT
JUMP TO SINE ROUTINE

1013 13016 062733 JSM FAD1

1014 13017 066762 JMP SIN+1

1015*

1016*****

1017 13020 000000 BSS 1

CHECKSUM WORD

1018*

1019*****

1020*

PAGE 192

FUNCTIONS FOR 9830

```

1022
1023*
1024*****
1025*
1026* FIND THE ARCCOSINE OF AN ARGUMENT
1027*
1028 13021 024132 ACS LDB N1 LET THE FUNCTION FLAG BE -1
1029 13022 063032 JSM ASNS+1 JUMP TO THE ARCSINE ROUTINE
1030 13023 021706 LDA GFG LOAD THE ORIGINAL EXPONENT WORD
1031 13024 070211 SLA ACSC SKIP IF THE MANTISSA SIGN BIT IS 0
1032 13025 062571 JSM ITR OTHERWISE FETCH ANSWER
1033 13026 026535 LDB PI
1034 13027 062733 JSM FADI
1035 13030 067256 ACSC JMP EATN JUMP TO CONVERT TO THE PROPER UNITS

1036*
1037*
1038*****
1039*
1040* SUBROUTINE FOR THE ARC ROUTINES
1041*
1042 13031 024012 ASNS LDB PI LET THE FUNCTION FLAG BE 1
1043 13032 035710 STB COUNT STORE THE FUNCTION FLAG
1044 13033 070537 LDB A,I SAVE THE EXPONENT WORD
1045 13034 035706 STB GFG FOR FUTURE REFERENCES
1046 13035 062550 JSM INIT+1 FETCH THE ARGUMENT AND INITIALIZE THE FLAGS
1047 13036 062653 JSM SCRT+1 CALCULATE SQR (1+OR-XX)
1048 13037 074070 SIB *+1 LOAD THE FIRST 4 DIGITS
1049 13040 074537 LDB B,I OF THE NUMBER SPECIFIED BY THE ADDRESS IN B
1050 13041 076210 RZB *+4 SKIP IF THEY ARE NOT ZERO
1051 13042 061256 JSM CLAR2 *CLEAR AR2
1052 13043 021716 LDA SIGN *LOAD A WITH THE ORIGINAL SIGN
1053 13044 067244 JMP POOT+3 JUMP TO ATNS ROUTINE TO STORE +- PI/2
1054*
1055 13045 062674 JSM SCRT1 JUMP TO FETCH THE DIVIDE ADDRESSES OF SCRT
1056 13046 062731 JSM FDV+1 DIVIDE THE RESULTS OF SCRT
1057 13047 021703 LDA ADR3 SET THE ADDRESS OF THE RESULT
1058 13050 067206 JMP ATNS JUMP TO THE ARCTANGENT ROUTINE

1059*
1060*
1061*****
1062*
1063* FIND THE ARCSINE OF AN ARGUMENT
1064*
1065 13051 063031 ASM JSM ASNS CALCULATE THE ANSWER IN RADIANS
1066 13052 067256 JMP EATN CONVERT IT TO THE PROPER UNITS

```


PAGE 193

FUNCTIONS FOR 9830

1068*

1069*

1070*****

1071*

1072* FIND THE TANGENT OF AN ARGUMENT

1073*

| | | | | | | |
|-------|-------|--------|-------|-----|--------|---|
| 1074 | 13053 | 062714 | TAN | JSM | STN | FETCH THE ARGUMENT IN
RADIANS AND INITIALIZE FLAGS |
| 1075 | 13054 | 024012 | | LDB | PI | SET THE RECIPROCAL |
| 1076 | 13055 | 035711 | | STB | RCPFG | FLAG TO 1 |
| 1077 | 13056 | 025754 | | LDB | AR2E | LOAD THE EXPONENT WORD |
| 1078 | 13057 | 035716 | | STB | SIGN | |
| 1079 | 13060 | 074153 | | SBP | *+3 | SKIP IF THE EXPONENT IS POSITIVE |
| 1080 | 13061 | 045706 | GET 3 | ISZ | GFG | OTHERWISE SET GFG=1 AND |
| 1081 | 13062 | 067116 | | JMP | GET | DO NOT SCALE |
| 1082 | 13063 | 026536 | | LDB | TWOPI | SCALE THE ARGUMENT TO |
| 1083 | 13064 | 062605 | | JSM | FRAC | WITHIN ONE CIRCLE |
| 1084 | 13065 | 062635 | | JSM | ZERO | DETERMINE IF THE RESULT IS
ZERO |
| 1085 | 13066 | 070635 | | SEC | FAL1,C | STORE ZERO IF IT IS |
| 1086 | 13067 | 075512 | | SBM | GET3 | SKIP IF THE EXPONENT IS NOW
NONZERO |
| 1087* | | | | | | |
| 1088 | 13070 | 020023 | | LDA | PI/2 | LOAD PI/2 |
| 1089 | 13071 | 062754 | | JSM | XAR1 | INTO AR1 |
| 1090 | 13072 | 170400 | | CMY | | COMPLEMENT IT |
| 1091 | 13073 | 074742 | | SBR | 16 | SUBTRACT AR1 FROM AR2 |
| 1092 | 13074 | 170420 | | FDV | | UNTIL AR2 GOES NEGATIVE |
| 1093 | 13075 | 935711 | | STB | RCPFG | STORE THE NO. OF SHIFTS
BEFORE OVERFLOW |
| 1094 | 13076 | 062635 | | JSM | ZERO | DETERMINE IF AR2 IS NOW ZERO |
| 1095 | 13077 | 070434 | | SES | *+8,C | SKIP IF IT IS NOT ZERO |
| 1096* | | | | | | |
| 1097 | 13100 | 025711 | | LDB | RCPFG | SKIP IF THE NO. OF SUBTRACTS
ABOVE WAS EVEN |
| 1098 | 13101 | 074111 | | SLB | *+2 | |
| 1099 | 13102 | 065256 | FAL1 | JMP | CLAR2 | STORE ZERO IF ODD — EXIT HERE
FOR SIN0, COS0, TAN |
| 1100 | | | | | | |
| 1101 | 13103 | 025754 | | LDB | AR2E | LOAD THE EXPONENT WORD |
| 1102 | 13104 | 055704 | | DSZ | TEMP | STORE MAX NO. IF THE SIN |
| 1103 | 13105 | 065274 | | JMP | STMAX | ROUTINE WAS NOT CALLED |
| 1104 | 13106 | 170402 | | RET | | OTHERWISE RETURN WITH A=0 |
| 1105* | | | | | | |
| 1106 | 13107 | 062576 | | JSM | NORM+1 | NORMALIZE AR2 |
| 1107* | | | | | | |
| 1108 | 13110 | 074312 | | SBM | GET | SKIP IF THE NEW EXPONENT IS
NEGATIVE |
| 1109 | 13111 | 020157 | | LDA | N256 | MAKE THE EXPONENT |
| 1110 | 13112 | 041716 | | IOR | SIGN | MINUS ONE AND |
| 1111 | 13113 | 031754 | | STA | AR2E | ORIGINAL MANTISSA SIGN BIT |
| 1112 | 13114 | 031716 | | STA | SIGN | STORE THIS ALSO OVER THE
ORIGINAL EXPONENT |
| 1113* | | | | | | |
| 1114 | 13115 | 067121 | | JMP | GET+3 | JUMP TO AVOID THE OFFSET
BELOW |
| 1115 | 13116 | 035716 | GET | STB | SIGN | STORE THE NEW EXPONENT WORD |
| 1116 | 13117 | 024012 | | LDB | PI | SET B=1 AND |
| 1117 | 13120 | 061305 | | JSM | ROUND | JUMP TO RIGHT SHIFT AR2 |
| 1118* | | | | | | |
| 1119 | 13121 | 063523 | | JSM | ATD | INITIALIZE THE DIGIT STACK
POINTER |
| 1120* | | | | | | |
| 1121 | 13122 | 021754 | | LDA | AR2E | LOAD THE EXPONENT WORD |
| 1122 | 13123 | 070240 | | AAR | 6 | MULTIPLY THE EXPONENT
WORD BY 4 |
| 1123 | 13124 | 070076 | | TCA | | COMPLEMENT IT |

PAGE 194

FUNCTIONS FOR 9830

| | | | | | |
|-------|-------|--------|-----|-------|---|
| 1124 | 13125 | 002546 | ADA | THETT | ADD IT TO THE TRIG TABLE LINK IN A |
| 1125 | 13126 | 025754 | LDB | AR2E | LOAD THE EXPONENT WORD |
| 1126 | 13127 | 074340 | ABR | 8 | ALIGN THE EXPONENT WORD |
| 1127 | 13130 | 004024 | ADB | P7 | AND ADD 7 TO IT |
| 1128 | 13131 | 074453 | SBP | .T. | SKIP IF THE RESULT IS POSITIVE |
| 1129* | | | | | |
| 1130 | 13132 | 171450 | NRM | | NORMALIZE AR2 |
| 1131 | 13133 | 062571 | JSM | ITR | TRANSFER AR2 INTO THE RESULT |
| 1132 | 13134 | 025711 | LDB | RCPFG | LOAD THE NO. OF SUBTRACTS FROM
LOCATION ABOVE |
| 1133 | 13135 | 074111 | SLB | *+2 | SKIP IF IT IS EVEN |
| 1134 | 13136 | 067200 | JMP | OUT-2 | IF ODD JUMP TO THE STORE ROUTINE |
| 1135 | 13137 | 062646 | JSM | RECIP | RECIPROCATATE THE RESULT |
| 1136 | 13140 | 025711 | LDB | RCPFG | LOAD THE NO. OF SUBTRACTS |
| 1137 | 13141 | 067200 | JMP | OUT-2 | JUMP TO THE STORE ROUTINE |
| 1138 | 13142 | 075410 | SZB | *+8 | SKIP BACK IF THE EXPONENT WAS
-7 |
| 1139* | | | | | |
| 1140 | 13143 | 063431 | JSM | IN | EXECUTE PHASE I |
| 1141* | | | | | |
| 1142 | 13144 | 021754 | LDA | AR2E | LOAD THE EXPONENT WORD |
| 1143 | 13145 | 000013 | ADA | P256 | INCREMENT THE EXPONENT BY 1 |
| 1144 | 13146 | 070340 | AAR | 8 | SHIFT TO THE LOW 8 BITS IN A |
| 1145 | 13147 | 074742 | SBR | 16 | SET THE EXPONENT |
| 1146 | 13150 | 035754 | STB | AR2E | WORD OF AR2 TO 0 |
| 1147 | 13151 | 070137 | LDB | A | SET 8 EQUAL TO 6 |
| 1148 | 13152 | 070076 | TCA | | MINUS THE |
| 1149 | 13153 | 004023 | ADB | P6 | 6 REGISTER |
| 1150 | 13154 | 063457 | JSM | IN2 | EXECUTE PHASE II |
| 1151* | | | | | |
| 1152 | 13155 | 021711 | LDA | RCPFG | LOAD THE RECIPROCAL FLAG |
| 1153 | 13156 | 070211 | SLA | *+4 | SKIP IF IT IS EVEN |
| 1154 | 13157 | 020213 | LDA | AR1A | SET A TO THE ADD. OF CORDIC X |
| 1155 | 13160 | 025713 | LDB | RY1 | SET B TO THE ADDRESS OF CORDIC
Y |
| 1156 | 13161 | 067164 | JMP | *+3 | JUMP TO NORMALIZE THESE |
| 1157 | 13162 | 021713 | LDA | RY1 | SET A TO THE ADDRESS OR CORDIC
Y |
| 1158 | 13162 | 024213 | LDB | AR1A | SET B TO THE ADD. OF CORDIC X |
| 1159 | 13164 | 062563 | JSM | DIV | JUMP TO NORMALIZE THESE |
| 1160 | 13165 | 022542 | LDA | RSLT | LOAD THE ADDRESS OF THE DIVISOR |
| 1161 | 13166 | 026541 | LDB | TREG | LOAD THE ADDRESS OF THE
DIVIDEND |
| 1162 | 13167 | 062730 | JSM | .FDV | JUMP TO THE DIVIDE ROUTINE |
| 1163* | | | | | |
| 1164 | 13170 | 025716 | LDB | SIGN | LOAD THE EXPONENT ORIGINAL |
| 1165 | 13171 | 005711 | ADB | RCPFG | ADD THE RECIPROCAL FLAG TO THE
EXPONENT WORD INTO A
MANTISSA SIGN BIT |
| 1166 | 13172 | 004013 | ADB | P256 | INCREMENT THE EXPONENT BY 1 |
| 1167 | 13173 | 021711 | LDA | RCPFG | LOAD THE RECIPROCAL FLAG INTO B |
| 1168 | 13174 | 070111 | SLA | *+2 | SKIP IF IT IS EVEN |
| 1169 | 13175 | 067200 | JMP | OUT-2 | JUMP TO THE STORE ROUTINE |
| 1170 | 13176 | 074076 | TCB | | COMPLEMENT B AND |
| 1171 | 13177 | 004033 | ADB | D0010 | PROPOGATE BIT 7 INTO THE
EXPONENT |
| 1172 | 13200 | 005706 | ADB | GFG | ADD THE G FLAG TO THE MANTISSA
SIGN INB |

310

298

1,444,141

298

PAGE 194 Continued FUNCTIONS FOR 9830

| | | | | | | |
|-------|-------|--------|-----|-----|-------|------------------------------|
| 1173 | 13201 | 021754 | | LDA | AR2E | STORE THE EXPONENT WORD OF |
| 1174 | 13202 | 031744 | OUT | STA | AR1E | AR2 |
| 1175* | | | | | | IN AR1 FOR THE STORE ROUTINE |
| 1176 | 13203 | 022542 | | LDA | RSLT | RESET THE RETURN |
| 1177 | 13204 | 031703 | | STA | ADR3 | LOCATION ADDRESS |
| 1178* | | | | | | |
| 1179 | 13205 | 065237 | ZOR | JMP | STAR2 | JUMP TO THE STORE ROUTINE |

(311)

299

1,444,141

299

PAGE 195

FUNCTIONS FOR 9830

```

1181*
1182*
1183*****
1184*
1185*  ARCTANGENT SUBROUTINE
1186*
1187  13206  024062  ATNS    LDB  D8000  LOAD THE ARCTANGENT FLAG —
                                         BCD 8000
1188  13207  062547          JSM  INIT   JUMP TO FETCH THE ARGUMENT
                                         AND INITIALIZE THE FLAGS
1189*
1190  13210  071655          SEC  ZOR     RETURN IF IT IS ZERO
1191  13211  074312          SBM  OK+1   SKIP IF THE EXPONENT IS NEGATIVE
1192  13212  006544          ADB  EN12   IF THE EXPONENT IS LESS THAN 12
1193  13213  074152          SBM  OK     JUMP TO CONTINUE
1194  13214  061256  FAP    JSM  CLAR2  OTHERWISE SET AR2=0 AND JUMP TO
1195  13215  067241          JMP  POOT   STORE AN ANSWER OF PI/2
1196*
1197  13216  062646  OK     JSM  RECIP   RECIPROCATATE THE ARGUMENT
1198*
1199  13217  062704          JSM  CLER   CLEAR THE DIGIT STACK
1200  13220  021754          LDA  AR2E   SET THE A REGISTER EQUAL
1201  13221  070300          AAR  7     TO 22-2* THE EXPONENT
1202  13222  070076          TCA          TO BE USED AS A SHIFT
1203  13223  000035          ADA  P12   FACTOR IN PHASE II
1204  13224  024030          LDB  P12   SET R=12 TO BE USED AS A LOOP COUNT
1205*
1206  13225  063457          JSM  IN2    EXECUTE PHASE II
1207  13226  021754          LDA  AR2E   SAVE THE EXPONENT
1208  13227  031714          STA  SAVE   OF AR2 IN SAVE
1209  13230  070340          AAR  8     SET A EQUAL TO
1210  13231  070076          TCA          6 MINUS THE EXPONENT
1211  13232  000023          ADA  P6     TO BE USED AS A LOOP COUNT
1212*
1213  13233  000144          ADA  N13    IN PHASE III
1214  13234  070112          SAM  *+2   IF THIS QUANTITY IS GREATER
1215  13235  020132          LDA  N1     THAN 12 SET THE
1216  13236  000031          ADA  P13    A REGISTER TO 12
1217  13237  026535          LDB  PI     LET B=THE ADDRESS OF THE TRIG
                                         CORDIC TABLE
1218  13240  063654          JSM  IN3    EXECUTE PHASE III
1219*
1220  13241  021716  POOT   LDA  SIGN   LOAD THE ORIGINAL EXPONENT
                                         WORD
1221  13242  031754          STA  AR2E   AND STORE IT IN AR2
1222  13243  070452          SAM  GONO   SKIP IF THE EXPONENT IS NEGATIVE
1223*
1224  13244  050012          AND  P1     OTHERWISE CLEAR ALL BUT THE
                                         MANTISSA
1225  13245  031754          STA  AR2E   SIGN BIT IN THE EXPONENT WORD
1226  13246  025714          LDB  SAVE   LOAD THE EXPONENT FROM
                                         AFTER THE RECIPROCATATION
1227  13247  062625          JSM  UNRM+1 UNNORMALIZE AR2
1228*
1229  13250  020023          LDA  PI/2   SET PI/2 IN THE
1230  13251  062754          JSM  XAR1   AR1 TEMPORARY
1231*
1232  13252  170400          CMY          CALCULATE
1233  13253  170560          FXA          AR2=AR1-AR2
1234*
1235  13254  066576  GONO   JMP  NORM+1 NORMALIZE THE RESULT
1236*

```

312

300

1,444,141

300

PAGE 196 FUNCTIONS FOR 9830

1237*

1238*****

1239*

1240* FIND THE ARCTANGENT OF AN ARGUMENT

1241*

1242 13255 063206 ATN JSM ATNS JUMP TO CALCULATE THE ANSWER IN RADIANS

1243*

1244 13256 025434 EATN LDB UNITS LOAD THE UNITS MODE WORD

1245 13257 076110 RZB *+2 IF IT IS SET TO DEGREES

1246 13260 062743 JSM RTOD CONVERT THE ANSWER TO DEGREES

1247 13261 022533 LDA GPERR IF GRADS ARE SET

1248 13262 014017 CPB GRA CONVERT THE ANSWER

1249 13263 062744 JSM RTOD+1 TO GRADS

1250*

1251 13264 074742 SOUT SBR 16 CLEAR B AND RETURN

1252 13265 067201 JMP OUT-1 THRU THE STORE ROUTINE

1253*

1254*

1255*****

PAGE 197 FUNCTIONS FOR 9830

1257* RAISE NAPERIAN E TO A POWER

| | | | | | |
|-------|-------|--------|-----|----------|-----------------------------------|
| 1258* | | | | | |
| 1259 | 13266 | 021703 | LDA | ADR2 | |
| 1260 | 13267 | 062751 | JSM | FMP | |
| 1261 | 13270 | 021703 | LDA | ADR3 | |
| 1262 | 13271 | 024012 | LDB | PI | LET THE FUNCTION FLAG BE 1 |
| 1263 | 13272 | 062547 | JSM | INIT | FETCH THE ARGUMENT AND |
| | | | | | INITIALIZE THE FLAGS |
| 1264 | 13273 | 006543 | ADB | EN3 | IF THE EXPONENT IS 3 OR GREATER |
| 1265 | 13274 | 074413 | SBP | STM1-1 | JUMP TO THE STORE MAX ROUTINE |
| 1266* | | | | | |
| 1267 | 13275 | 026534 | LDB | LN10 | SUBTRACT AS MANY WHOLE LN10'S |
| 1268 | 13276 | 062605 | JSM | FRAC | FROM THE NO. AS POSSIBLE |
| 1269 | 13277 | 062624 | JSM | UNRM | UNNORMALIZE THE RESULT |
| 1270* | | | | | |
| 1271 | 13300 | 021710 | LDA | COUNT | LOAD THE NO. OF SUBTRACTS FROM |
| | | | | | ABOVE |
| 1272 | 13301 | 070542 | SAR | 12 | ISOLATE THE FIRST DIGIT |
| 1273 | 13302 | 070210 | SZA | STM1+1 | IF IT IS NOT ZERO |
| 1274 | 13303 | 025716 | LDB | SIGN | LOAD THE EXPWORD OF THE ORIGINAL |
| | | | | | ARGUMENT |
| 1275 | 13304 | 074044 | SBL | 15 | POSITION THE SIGN BIT AND JUMP TO |
| 1276 | 13305 | 065264 | JMP | OF+3 | THE OVERFLOW/UNDERFLOW |
| | | | | | ROUTINE |
| 1277* | | | | | |
| 1278 | 13306 | 025710 | LDB | COUNT | LOAD THE NO. OF LN10'S |
| 1279 | 13307 | 074404 | SBL | 8 | ISOLATE THE UNITS DIGIT |
| 1280 | 13310 | 074142 | SBR | 4 | OF IT IN B |
| 1281 | 13311 | 021710 | LDA | COUNT | ISOLATE THE TENS DIGIT |
| 1282 | 13312 | 070342 | SAR | 8 | OF IT IN A |
| 1283 | 13313 | 070344 | SAL | 9 | MULTIPLY THE A REGISTER |
| 1284 | 13314 | 070037 | ADB | A | BY 10 AND ADD IT |
| 1285 | 13315 | 070704 | SAL | 2 | TO B |
| 1286 | 13316 | 070037 | ADB | A | THIS QUANTITY WILL BE THE |
| | | | | | EXPONENT |
| 1287 | 13317 | 035711 | STB | RCPFG | OF THE UNNORMALIZED ANSWER |
| 1288 | | | | | |
| 1289 | 13320 | 062704 | JSM | CLER | CLEAR THE DIGIT STACK |
| 1290 | 13321 | 063523 | JSM | ATD | INITIALIZE THE DIGIT |
| 1291 | 13322 | 055717 | DSZ | DIGIT | STACK POINTER |
| 1292 | 13323 | 022545 | LDA | THETL | PUT THE ADDRESS OF THE LOG |
| | | | | | CONSTANT STACK IN A |
| 1293 | 13324 | 024023 | LDR | P6 | SET B=6 LOOPS |
| 1294 | 13325 | 063431 | JSM | IN | EXECUTE PHASE I |
| 1295* | | | | | |
| 1296 | 13326 | 024136 | LDB | N5 | TRANSFER THE LAST |
| 1297 | 13327 | 045717 | ISZ | DIGIT | SIX DIGITS OF THE |
| 1298 | 13330 | 171400 | MLS | | REDUCED ARGUMENT |
| 1299 | 13331 | 131717 | STA | DIGIT, I | INTO THE DIGIT STACK |
| 1300 | 13332 | 077670 | RIB | *-3 | FROM AR2 |
| 1301* | | | | | |
| 1302 | 13333 | 062753 | JSM | OTO | SET AR1=1 |
| 1303 | 13334 | 063603 | JSM | PTJ | EXECUTE PHASE IV |
| 1304* | | | | | |
| 1305 | 13335 | 021711 | LDA | RCPFG | STORE THE EXPONENT CALCULATED |
| 1306 | 13336 | 031754 | STA | AR2F | ABOVE IN AR2 |
| 1307 | 13337 | 171450 | NRM | | NORMALIZE THE ANSWER |
| 1308* | | | | | |
| 1309 | 13340 | 021716 | LDA | SIGN | IF THE ORIGINAL ARGUMENT |
| 1310 | 13341 | 070111 | SLA | *+2 | WAS NEGATIVE RECIPROCATATE |
| 1311 | 13342 | 062646 | JSM | RECIP | THE ANSWER |
| 1312 | 13343 | 067264 | JMP | SOUT | JUMP TO THE STORE ROUTINE |

314

302

1,444,141

302

PAGE 198

FUNCTIONS FOR 9830

```

1314*
1315*
1316*****
1317*
1318* FIND THE NATURAL LOGARITHM OF AN ARGUMENT
1319*
1320 13344 024132 LNX LDB N1 SET THE FUNCTION FLAG TO -1
1321 13345 062547 JSM INIT FETCH THE ARGUMENT AND
INITIALIZE THE FLAGS
1322*
1323 13346 070314 SES *+6
1324 13347 160205 JSM AERRA, I
1325 13350 177632 DEC -102
1326 13351 024012 LDB P1 SET THE FUNCTION FLAG TO 1
1327 13352 035712 STB FNCFG FNCFG IS FLAGED TO RETURN
-9.9E99 WHEN LOG10 CALL.
1328 13353 065274 JMP STMAX
1329 13354 074111 SLB *+2
1330 13355 160206 ER47 JSM ERRA, I
1331*
1332 13356 062704 JSM CLER CLEAR THE DIGIT STACK
1333 13357 063603 JSM PTJ EXECUTE PHASE IV
1334 13360 055717 DSZ DIGIT DECREMENT THE DIGIT STACK
POINTER
1335*
1336 13361 020023 LDA B6 LET I=6 LOOPS
1337 13362 026534 LDB LN10 LET B=THE ADDRESS OF THE LOG
CONSTANT STACK
1338*
1339 13363 063654 JSM IN3 EXECUTE PHASE III
1340*
1341 13364 026540 LDB ADR0 SAVE THE MANTISSA OF THE ANSWER
IN
1342 13365 020214 LDA AR2A IN A TEMPORARY
1343 13366 170004 XFR
1344*
1345 13367 062757 JSM OTT SET AR2=1
1346*
1347 13370 021716 LDA SIGN LOAD THE ORIGINAL EXPONENT
1348 13371 000013 ADA P256 INCREMENT IT
1349 13372 070153 SAP *+3 IF IT IS NEGATIVE
1350 13373 070071 SLA *+C CLEAR THE MANTISSA SIGN
1351 13374 070076 TCA AND COMPLEMENT IT
1352 13375 070342 SAR 8 SHIFT IT TO THE LOW 8 BITS
1353*
1354 13376 000156 ADA N100 SUBTRACT 100 FROM IT
1355 13377 024013 LDB D0100 SET B=RCD 0100
1356 13400 070153 SAP *+3 SKIP IF THE EXPONENT IS POSITIVE
1357 13401 000060 ADA P100 OTHERWISE RESTORE IT AND
1358 13402 074742 SBR 16 CLEAR B
1359*
1360 13403 000141 ADA N10 SUBTRACT 10 FROM IT
1361 13404 070152 SAM *+3 SKIP IF IT IS NEGATIVE
1362 13405 004033 ADB D0010 OTHERWISE ADD BCD 0010 TO B
1363 13406 067403 JMP *-3 JUMP BACK TO ADD AGAIN
1364 13407 000061 ADA P10 RESTORE A
1365*
1366 13410 070037 ADB A ADD THE ONES LEFT TO B
1367 13411 035755 STB AR2E+1 STORE THIS WORD AS THE FIRST 4
DIGITS OF AR2
1368*
1369 13412 020110 LDA EP3 SET THE EXPONENT

```

PAGE 199

FUNCTIONS FOR 9830

| | | | | | |
|-------|-------|--------|-----|--------|------------------------------|
| 1370 | 13413 | 031754 | STA | AR2E | OF AR2 TO +3 |
| 1371 | 13414 | 062576 | JSM | NORM+1 | NORMALIZE AR2 |
| 1372* | | | | | |
| 1373 | 13415 | 022534 | LDA | LN10 | MULTIPLY AR2 BY |
| 1374 | 13416 | 062750 | JSM | .FMP-1 | LOG OF 10 |
| 1375* | | | | | |
| 1376 | 13417 | 021703 | LDA | ADR3 | NORMALIZE THIS AND THE |
| 1377* | | | | | |
| 1378 | 13420 | 026540 | LDB | ADR0 | RESULT OF THE |
| 1379 | 13421 | 062563 | JSM | DIV | CORDIC ITERATION |
| 1380* | | | | | |
| 1381 | 13422 | 146542 | ISZ | RSLT,I | NEGATE THE MANTISSA QUANTITY |
| 1382* | | | | | |
| 1383 | 13423 | 021716 | LDA | SIGN | IF THE EXPONENT OF THE |
| 1384 | 13424 | 070113 | SAP | *+2 | ORIGINAL WAS NEGATIVE |
| 1385 | 13425 | 045630 | ISZ | TREG | NEGATE THE LN10 MULTIPLE |
| 1386 | 13426 | 022541 | LDA | TREG | ADD THESE TWO |
| 1387 | 13427 | 026542 | LDB | RSLT | QUANTITIES FOR THE |
| 1388 | 13430 | 066734 | JMP | .FAD | FINAL ANSWER |

316

304

1,444,141

304

PAGE 200

FUNCTIONS FOR 9830

```

1390*
1391*
1392*****
1393*
1394*  CORDIC LOOP
1395*
1396*
1397*****
1398*
1399*  PHASE I
1400*
1401  13431  031705  IN      STA  TABL0  STORE THE TABLE ADDRESS
1402  13432  035707      STB  SHIFT  STORE THE NO. OF LOOPS
1403*
1404  13433  062635  PTA     JSM  ZERO   DETERMINE IF THE ARGUMENT IS ZERO
1405  13434  070154      SES  *+3    IF IT IS INCREMENT THE
1406  13435  145717      ISZ  DIGIT,I LAST DIGIT IN THE STACK
1407  13436  170402      RET                    AND RETURN
1408*
1409  13437  170400      CMY                    COMPLEMENT THE ARGUMENT
1410*
1411  13440  045717      ISZ  DIGIT  INCREMENT THE DIGIT STACK POINTER
1412*
1413  13441  020021      LDA  P4      FETCH THE NEXT CONSTANT
1414  13442  001705      ADA  TABL0  FETCH THE NEXT CONSTANT
1415  13443  031705      STA  TABL0  INTO AR1
1416  13444  024213      LDB  AR1A
1417  13445  170004      XFR
1418*
1419  13446  074004      SBL  16      ADD AR1 TO AR2
1420  13447  170420      FDV                    UNTIL OVERFLOW
1421*
1422  13450  135717      STB  DIGIT,I STORE THE NO. OF SUBTRACTS BEFORE
                                           OVERFLOW
1423*
1424  13451  170400      CMY                    RESTORE THE
1425  13452  170560      FXA                    ARGUMENT TO ITS LAST NEGATIVE
                                           VALUE
1426*
1427  13453  171400      MLS                    LEFT SHIFT THE ARGUMENT
1428*
1429  13454  055707      DSZ  SHIFT  DECREMENT THE LOOP COUNTER
1430  13455  067433      JMP  PTA   CONTINUE IF IT IS NONZERO
1431  13456  170402      RET                    OTHERWISE RETURN

```

PAGE 201

FUNCTIONS FOR 9830

1433*
 1434*
 1435*****
 1436*
 1437* PHASE II
 1438*
 1439 13457 031714 IN2 STA SAVE STORE THE SHIFT FACTOR
 1440 13460 035707 STB SHIFT STORE THE LOOP COUNT
 1441 13461 062753 JSM OTO SET AR1=1
 1442*
 1443 13462 022540 LDA ADRY INITIALIZE RY1 AND RY2
 1444 13463 031713 STA RY1 TO THE ADDRESSES OF
 1445 13464 020111 LDA ADRX TWO TEMPORARIES FOR
 1446 13465 031715 STA RY2 INTERMEDIATE STORAGE
 1447*
 1448 13466 025712 LDB FNCFG SKIP TO PTE IF THE ATN
 1449 13467 076650 RZB PTE IS BEING CALCULATED
 1450 13470 067525 JMP PTL OTHERWISE GO TO PTL
 1451*
 1452 13471 076710 PTB RZB PTN
 1453 13472 067534 JMP PTC
 1454*
 1455 13473 055717 PTD DSZ DIGIT DECREMENT THE DIGIT STACK POINTER
 1456 13474 025712 LDB FNCFG SKIP TO PTL IF THE
 1457 13475 075550 SZB PTL1 TAN IS BEING CALCULATED
 1458*
 1459 13476 021755 PTO LDA AR2E+1 LOAD THE FIRST FOUR DIGITS OF AR2
 1460 13477 070542 SAR 12 ISOLATE THE FIRST ONE
 1461 13500 070150 SZA *+3 AND SKIP IF IT IS ZERO
 1462 13501 170410 DRS OTHERWISE RIGHT SHIFT
 1463 13502 067507 JMP PTN AR2 AND JUMP TO PTN
 1464 13503 171400 MLS LEFT SHIFT AR2 ONCE
 1465 13504 020214 PTE LDA AR2A STORE THE NEW VALUE
 1466 13505 025713 LDB RY1 OF Y IN
 1467 13506 170004 XFR RY1
 1468*
 1469 13507 174400 PTN CMX NEGATE AR1=X
 1470 13510 170560 FXA AR2=AR2-AR1 (Y-X)
 1471 13511 070314 SES *+6 SKIP ON OVERFLOW
 1472 13512 021713 LDA RY1 OTHERWISE RESTORE
 1473 13513 024214 LDB AR2A AR2 WITH ITS
 1474 13514 170004 XFR LAST VALUE
 1475 13515 174400 CMX RECOMPLEMENT AR1=X
 1476 13516 067600 PTF1 JMP PTF JUMP TO PTF
 1477*
 1478 13517 174400 CMX RECOMPLEMENT AR1
 1479*
 1480 13520 062635 JSM ZERO DETERMINE IF AR2=0
 1481 13521 070634 SES PTM,C SKIP IF IT IS NOT
 1482 13522 155717 DSZ DIGIT,1 IF IT IS ZERO DECREMENT THE
 1483 13523 022537 ATD LDA AWRDN LAST DIGIT AND FINALIZE THE
 1484 13524 066712 JMP CK-1 DIGIT STACK POINTER AND RETURN
 1485 13525 024012 PTL LDB PI RIGHT SHIFT AND ROUND UP
 1486 13526 061305 JSM ROUND AR1 ONCE
 1487*
 1488 13527 020214 LDA AR2A STORE THIS

318

306

1,444,141

306

PAGE 202

FUNCTIONS FOR 9830

| | | | | | |
|-------|-------|--------|-----|-------------|---|
| 1489 | 13530 | 025713 | | LDB RY1 | NEW Y IN |
| 1490 | 13531 | 170004 | | XFR | RY1 |
| 1491* | | | | | |
| 1492 | 13532 | 121717 | | LDA DIGIT,I | IF THE PRESENT DIGIT IS ZERO FOR |
| 1493 | 13533 | 071150 | | SZA PTF1 | THE TAN JUMP TO PTF |
| 1494* | | | | | |
| 1495 | 13534 | 170560 | PTC | FXA | OTHERWISE CALCULTE AR2=AR2+ARY
(Y=Y+X) |
| 1496* | | | | | |
| 1497 | 13535 | 021713 | PTM | LDA RY1 | EXCHANGE THE NEW |
| 1498 | 13536 | 025715 | | LDB RY2 | AND OLD VALUES |
| 1499 | 13537 | 031715 | | STA RY2 | OF Y BY EXCHANGING |
| 1500 | 13540 | 035713 | | STB RY1 | THEIR ADDRESSES |
| 1501 | 13541 | 020214 | | LDA AR2A | STORE THE NEW |
| 1502 | 13542 | 170004 | | XFR | Y INTO AR2 |
| 1503* | | | | | |
| 1504 | 13543 | 021715 | | LDA RY2 | FETCH THE VALUE |
| 1505 | 13544 | 024214 | | LDB AR2A | OF Y USED IN THE LAST LOOP |
| 1506 | 13545 | 170004 | | XFR | |
| 1507* | | | | | |
| 1508 | 13546 | 021712 | | LDA FNCFG | LOAD THE FUNCTION FLAG |
| 1509 | 13547 | 025707 | | LDB SHIFT | LOAD THE LOOP COUNTER |
| 1510 | 13550 | 004132 | | ADB N1 | SUBTRACT 1 FROM IT |
| 1511 | 13551 | 074774 | | SBL 1 | MULTIPLY IT BY 2 |
| 1512* | | | | | |
| 1513 | 13552 | 070150 | | SZA *+3 | SKIP IF TAN |
| 1514 | 13553 | 074076 | | TCB | COMPLEMENT IT AND |
| 1515 | 13554 | 067556 | | JMP *+2 | JUMP |
| 1516 | 13555 | 005714 | | ADB SAVE | ADD THE SHIFT FACTOR TO IT |
| 1517 | 13556 | 005714 | | ADB SAVE | TWICE IF TAN AND ONCE IF ATN |
| 1518* | | | | | |
| 1519 | 13557 | 074117 | | LDA B | IF THE RESULTING |
| 1520 | 13560 | 000143 | | ADA N12 | VALUE IS 12 OR |
| 1521 | 13561 | 070453 | | SAP PTK | MORE SKIP TO PTK |
| 1522* | | | | | |
| 1523 | 13562 | 061305 | | JSM ROUND | RIGHT SHIFT AR2 B TIMES AND
ROUND UP |
| 1524* | | | | | |
| 1525 | 13563 | 025712 | | LDB FNCFG | LOAD THE FUNCTION FLAG |
| 1526 | 13564 | 076110 | | RZB *+2 | AND COMPLEMENT AR2 IF |
| 1527 | 13565 | 170400 | | CMY | THE TAN IS BEING CALCULATED |
| 1528 | 13566 | 170560 | | FXA | ADD THE SHIFTED Y TO X FOR A NEW X |
| 1529* | | | | | |
| 1530 | 13567 | 020214 | | LDA AR2A | STORE THE NEW VALUE OF X |
| 1531 | 13570 | 024213 | | LDB AR1A | BACK IN AR1 |
| 1532 | 13571 | 170004 | | XFR | |
| 1533* | | | | | |
| 1534 | 13572 | 021713 | PTK | LDA RY1 | FETCH THE LAST |
| 1535 | 13573 | 024214 | | LDB AR2A | CALCULATED VALUE OF Y INTO AR2 |
| 1536 | 13574 | 170004 | | XFR | |
| 1537* | | | | | |
| 1538 | 13575 | 025712 | | LDB FNCFG | LOAD THE FUNCTION FLAG |
| 1539 | 13576 | 155717 | | DSZ DIGIT,I | DECREMENT THE PRESENT DIGIT |
| 1540 | 13577 | 067471 | | JMP PTB | JUMP TO PTB IF IT IS NOT ZERO |
| 1541* | | | | | |
| 1542 | 13600 | 055707 | PTF | DSZ SHIFT | OTHERWISE DECREMENT THE LOOP
COUNTER |
| 1543 | 13601 | 067473 | | JMP PTD | JUMP TO PTD IF NOT FINISHED |
| 1544 | 13602 | 170402 | | RET | OTHERWISE RETURN |

PAGE 203

FUNCTIONS FOR 9830

```

1546*
1547*
1548*****
1549*
1550* PHASE III
1551*
1552 13603 022537 PTJ LDA AWRDN INITIALIZE THE
1553 13604 031717 STA DIGIT DIGIT-STACK POINTER
1554*
1555 13605 020030 LDA P12 INITIALIZE THE
1556 13606 031707 STA SHIFT LOOP COUNT TO 12
1557 13607 025712 LDB FNCFG JUMP TO PTS IF THE
1558 13610 074312 SBM PTS LN IS BEING CALCULATED
1559 13611 062757 JSM OTT DUPLICATE AR1 IN AR2
1560*
1561 13612 025712 PTR LDB FNCFG JUMP TO PTI IF THE
1562 13613 074353 SBP PTI EXP IS BEING CALCULATED
1563 13614 174400 CMX SUBTRACT
1564 13615 170560 FXA AR1 FROM AR2
1565 13616 020214 PTS LDA AR2A DUPLICATE AR2, THE NEW X, IN AR1
1566 13617 024213 LDB AR1A
1567 13620 170004 XFR
1568 13621 067624 JMP PTH-2 JUMP
1569*
1570 13622 121717 PTI LDA DIGIT,1 SKIP IF THE PRESENT
1571 13623 070750 SZA GOON-1 DIGIT IN THE STACK IS ZERO
1572*
1573 13624 020061 LDA P10 INITIALIZE THE INNER
1574 13625 031705 STA TABL0 LOOP COUNT TO 10
1575 13626 025707 PTH LDB SHIFT SUBTRACT THE OUTER
1576 13627 074076 TCB LOOP COUNT FROM 12
1577 13630 004030 ADB P12 FOR THE SHIFT FACTOR
1578 13631 070742 SAR 16 RIGHT SHIFT AR1
1579 13632 174430 MRX THIS MANY TIMES
1580*
1581 13633 025712 LDB FNCFG IF THE EXP IS BEING CALCULATED
1582 13634 014012 CPB P1 SET THE OVERFLOW BIT IF THE
1583 13635 000136 ADA N5 LAST SHIFTED DIGIT IS >=5
1584*
1585 13636 170560 FXA CALCULATE THE NEW X
1586*
1587 13637 070454 SES PTG SKIP ON OVERFLOW
1588 13640 055705 DSZ TABL0 OTHERWISE DECREMENT THE INNER
LOOP COUNTER
1589 13641 067643 JMP GOON JUMP BACK IF THIS DIGIT COUNT
1590 13642 067650 JMP PTG IS LESS THAN 10
1591*
1592 13643 020214 GOON LDA AR2A DUPLICATE THE NEW VALUE
1593 13644 024213 LDB AR1A OF X IN AR1
1594 13645 170004 XFR
1595*
1596 13646 155717 DSZ DIGIT,1 DECREMENT THE STACK DIGIT
1597 13647 067626 JMP PTH JUMP TO PTN IF IT IS NOT ZERO
1598 13650 045717 PTG ISZ DIGIT OTHERWISE INCREMENT THE DIGIT
STACK POINTER
1599 13651 055707 DSZ SHIFT DECREMENT THE MAIN LOOP COUNTER
1600 13652 067612 JMP PTR JUMP TO PTR IF NOT FINISHED
1601 13653 170402 RET OTHERWISE RETURN

```

PAGE 204

FUNCTIONS FOR 9830

1603*
 1604*
 1605*****
 1606*
 1607* PHASE IV
 1608*
 1609 13654 035705 IN3 STB TABL0 STORE THE CONSTANT TABLE ADDRESS
 1610 13655 031704 STA TEMP STORE THE SHIFT FACTOR
 1611 13656 031707 STA SHIFT STORE THE LOOP COUNT
 1612*
 1613 13657 020214 LDA AR2A
 1614 13660 170000 CLR CLEAR AR2
 1615*
 1616 13661 121717 PTP LDA DIGIT,I LOAD THE PRESENT DIGIT FROM THE
 STACK
 1617 13662 070076 TCA COMPLEMENT IT
 1618 13663 024012 LDB P1 SHIFT IT INTO
 1619 13664 174470 MRY AR2
 1620 13665 025712 LDB FNCFG LOAD THE FUNCTION FLAG
 1621 13666 074151 SLB *+3 SKIP IF LN
 1622 13667 055717 DSZ DIGIT ESTABLISH WHERE THE NEXT DIGIT
 1623 13670 067672 JMP *+2
 1624 13671 045717 ISZ DIGIT IS
 1625 13672 055707 DSZ SHIFT DECREMENT THE LOOP COUNTER
 1626 13673 067661 JMP PTP JUMP BACK IF IT IS NONZERO
 1627*
 1628 13674 070742 SAR 16 ESTABLISH THE NEW LOOP
 1629 13675 025704 LDB TEMP COUNTER AS 12
 1630 13676 074076 TCB MINUS THE LAST
 1631 13677 004030 ADB P12 INITIAL ONE AND
 1632 13700 035707 STB SHFT SHIFT AR2 RIGHT
 1633 13701 174470 MRY THIS MANY TIMES
 1634*
 1635 13702 020135 PTP LDA N4 FETCH THE NEXT CONSTANT
 1636 13703 001705 ADA TABL0 INTO AR1
 1637 13704 031705 STA TABL0
 1638 13705 024213 LDB AR1A
 1639 13706 170004 XFR
 1640 13707 070742 SAR 16 CLEAR A
 1641 13710 025707 LDB SHIFT SKIP IF THE LOOP COUNT
 1642 13711 076110 RZB *+2 IS NOT ZERO
 1643 13712 170402 RET OTHERWISE RETURN
 1644 13713 004132 ADB N1 SHIFT AR1 THE LOOP COUNT
 1645 13714 174430 MRX MINUS 1 TIMES
 1646*
 1647 13715 000136 ADA N5 IF THE LAST SHIFTED
 1648 13716 070175 SEC *+3,C OUT DIGIT IS=75
 1649 13717 020213 LDA AR1A INCREMENT THE
 1650 13720 170540 MDI CONSTANT IN AR1
 1651*
 1652 13721 125717 LDB DIGIT,I LOAD THE PRESENT DIGIT
 1653 13722 074076 TCB COMPLEMENT IT
 1654 13723 171460 FMP ADD AR1 INTO AR2 THIS MANY TIMES
 1655*
 1656 13724 025712 LDB FNCFG
 1657 13725 074151 SLB *+3
 1658 13726 055717 DSZ DIGIT

PAGE 205

FUNCTIONS FOR 9830

| | | | | | |
|-------|--------------------------------------|--------|-----|---------|---|
| 1659 | 13727 | 067731 | JMP | *+2 | |
| 1660 | 13730 | 045717 | ISZ | DIGIT | |
| 1661 | 13731 | 055707 | DSZ | SHIFT | DECREMENT THE LOOP COUNTER |
| 1662 | 13732 | 067702 | JMP | PTQ | JUMP BACK IF IT IS NOT ZERO |
| 1663 | 13733 | 170402 | RET | | OTHERWISE RETURN |
| 1664* | | | | | |
| 1665* | ***** | | | | |
| 1666* | | | | | |
| 1667* | FIND THE LOG BASE TEN OF AN ARGUMENT | | | | |
| 1668* | | | | | |
| 1669 | 13734 | 063344 | LOG | JSM LNX | TAKE THE NATURAL LOG OF THE
ARGUMENT |
| 1670 | 13735 | 026534 | LDB | LN10 | |
| 1671 | 13736 | 055712 | DSZ | FNCFG | ZERO DIVIDE THE RESULT |
| 1672 | 13737 | 066727 | JMP | DTOR+1 | BY LN10 |
| 1673 | 13740 | 170402 | RET | | OTHERWISE RETURN WITH THE MAX NO. |
| 1674* | | | | | |
| 1675* | LAST WORD AVAILABLE 13740B | | | | |

PAGE 206

MATH ROUTINED FOR 9830, CALLED 'MATH'

```

1678*
1679* MATH ROUTINES BEGIN HERE
1680*
1681*
1682 11465          ORG 11465B
1683*
1684 11465 077764 MASK2 OCT 77764
1685*
1686*****
1687*
1688* FLOATING POINT SUBTRACT
1689*
1690*
1691 11466 031703 FSB STA ADR2
1692 11467 074117 LDA B
1693 11470 024057 LDB ATEMP
1694 11471 035702 STB ADR1
1695 11472 170004 XFR
1696 11473 120057 LDA ATEMP,I
1697 11474 072111 SLA *+2,S * CHANGE THE SIGN
1698 11475 070071 SLA *+1,C * AND ADD
1699 11476 130057 STA ATEMP,I
1700 11477 067502 JMP FAD+2
1701*
1702*
1703*****
1704*
1705* FLOATING POINT ADDITION
1706*
1707*
1708 11500 031703 FAD STA ADR2
1709 11501 035702 STB ADR1
1710 11502 125702 LDB ADR1,I "SUBTRACT" ENTRY POINT
1711 11503 074340 ABR 8 SUBTRACT THE
1712 11504 074076 TCB EXPONENT OF THE
1713 11505 121703 LDA ADR2,I SECOND ARGUMENT FROM
1714 11506 070340 AAR 8 THE FIRST ARGUMENT
1715 11507 070037 ADB A AND STORE THIS OFFSET
1716 11510 035701 STB TEMPA IN A TEMPORARY
1717*
1718 11511 021703 LDA ADR2 LOAD THE ADDRESS OF THE FIRST
1719 11512 074253 SBP *+5 ARGUMENT
1720 11513 024214 LDB AR2A JUMP IF THE 1ST AR1 IS LARGER
1721 11514 170004 XFR TRANSFER THE SMALLER NUMBER
1722 11515 024213 LDB AR1A INTO AR2
1723 11516 067522 JMP *+4 LOAD THE ADDRESS OF
1724 11517 024213 LDB AR1A AR1 AND JUMP
1725 11520 170004 XFR TRANSFER THE LARGER NUMBER
1726 11521 024214 LDB AR2A INTO AR1
1727 11522 021702 LDA ADR1 TRANSFER THE OTHER
1728 11523 170004 XFR ARGUMENT INTO THE
1729* OTHER WORKING REGISTER
1730*
1731 11524 021755 LDA AR2E+1 JUMP IF THE SMALLER
1732 11525 070510 SZA BNZ ARGUMENT IS ZERO TO STORE THE
1733* LARGER

```

PAGE 207

MATH ROUTINED FOR 9830, CALLED 'MATH'

| | | | | | | |
|-------|-------|--------|-------|-----|--------|---|
| 1734 | 11526 | 021745 | | LDA | AR1E+1 | JUMP IF THE LARGER |
| 1735 | 11527 | 072110 | | RZA | *+2 | NUMBER IS NON-ZERO |
| 1736 | 11530 | 065254 | | JMP | ZONK | STORE THE LARGER ARGUMENT |
| 1737* | | | | | | |
| 1738* | | | | | | |
| 1739 | 11531 | 021701 | | LDA | TEMPA | LOAD THE OFFSET |
| 1740 | 11532 | 070113 | | SAP | *+2 | AND COMPLEMENT IT |
| 1741 | 11533 | 070076 | | TCA | | IF ITS NEGATIVE |
| 1742 | 11534 | 070137 | | LDB | A | IF THE OFFSET IS GREATER |
| 1743 | 11535 | 003465 | | ADA | MASK2 | THAN ELEVEN THEN |
| 1744 | 11536 | 070253 | | SAP | *+5 | JUMP TO STORE THE |
| 1745 | 11537 | 020213 | BNZ | LDA | AR1A | LARGER NUMBER |
| 1746 | 11540 | 024214 | | LDB | AR2A | |
| 1747 | 11541 | 170004 | | XFR | | |
| 1748 | 11542 | 065254 | | JMP | ZONK | |
| 1749* | | | | | | |
| 1750* | | | | | | |
| 1751 | 11543 | 061305 | | JSM | ROUND | RIGHT JUSTIFY AND ROUND THE
SMALLER ARGUMENT |
| 1752* | | | | | | |
| 1753 | 11544 | 021744 | | LDA | AR1E | ADD THE SIGNS OF THE |
| 1754 | 11545 | 001754 | | ADA | AR2E | TWO MANTISSAS AND |
| 1755 | 11546 | 070511 | | SLA | INTLK | JUMP IF SAME |
| 1756* | | | | | | |
| 1757* | | | | | | |
| 1758* | | | | | | |
| 1759 | 11547 | 170400 | | CMY | | THEY HAVE ULIKE SIGNS SO
COMPLEMENT THE LARGER |
| 1760 | 11550 | 170560 | | FXA | | ADD THE MANTISSAS |
| 1761 | 11551 | 070154 | | SES | *+3 | IF A CARRY DIGIT IS GENERATED |
| 1762 | 11552 | 170400 | | CMY | | COMPLEMENT THE ANSWER AND |
| 1763 | 11553 | 045744 | | ISZ | AR1E | CHANGE THE SIGN OF THE LARGER
ARGUMENT |
| 1764 | 11554 | 171450 | | NRM | | NORMALIZE THE ANSWER. PUT THE
NO. OF SHIFTS IN B |
| 1765 | 11555 | 074076 | | TCB | | COMPLEMENT AND ALLIGN THE |
| 1766 | 11556 | 074404 | | SBL | 8 | NO. OF SHIFTS |
| 1767 | 11557 | 067572 | | JMP | .FAD. | JUMP TO STORE ROUTINE IF NOT ZERO |
| 1768* | | | | | | |
| 1769 | 11560 | 070074 | INTLK | SES | *+1,C | CLEAR THE E REGISTER |
| 1770 | 11561 | 170560 | | FXA | | ADD THE MANTISSAS |
| 1771 | 11562 | 070214 | | SES | *+4 | IF NO CARRY IS GENERATED |
| 1772 | 11563 | 021744 | | LDA | AR1E | LET THE EXPONENT WORD OF THE |
| 1773 | 11564 | 031754 | | STA | AR2E | LARGER ARGUMENT |
| 1774 | 11565 | 065254 | | JMP | ZONK | JUMP TO THE STORE ROUTINE |
| 1775 | 11566 | 170002 | | MOV | | ELSE SHIFT A ONE INTO |
| 1776 | 11567 | 070137 | | LDB | A | THE MOST SIGNIFICANT |
| 1777 | 11570 | 174470 | | MRY | | LOCATION OF THE ANSWER |
| 1778 | 11571 | 024013 | | LDB | P256 | LOAD B WITH THE EXPONENT
CORRECTION |
| 1779* | | | | | | |
| 1780 | 11572 | 021755 | .FAD. | LDA | AR2E+1 | |
| 1781 | 11573 | 070110 | | SZA | *+2 | IF THE MANTISSA WAS 0 CLEAR AR2 |
| 1782 | 11574 | 065237 | | JMP | STAR2 | |
| 1783 | 11575 | 065256 | | JMP | CLAR2 | |

PAGE 208

MATH ROUTINED FOR 9830, CALLED 'MATH'

```

1785*
1786*
1787*****
1788*
1789*   FLOATING POINT DIVISION
1790*
1791*
1792   11576  035702  FDV   STB  ADR1
1793   11577  031703      STA  ADR2
1794*
1795   11600  024214      LDB  AR2A   PUT THE DIVIDEND INTO AR2
1796   11601  170004      XFR
1797*
1798   11602  021702      LDA  ADR1   PLACE THE DIVISOR INTO AR1
1799   11603  024213      LDB  AR1A
1800   11604  170004      XFR
1801*
1802   11605  020057      LDA  ATEMP  INITIALIZE THE WORD
1803   11606  031475      STA  TEMP2  POINTER FOR THE QUOTIENT
1804   11607  170000      CLR        CLEAR THE QUOTIENT REGISTER
1805*
1806*
1807*
1808   11610  021745      LDA  AR1E+1
1809   11611  072310      RZA  GOOD
1810   11612  160205      JSM  AERRA,I
1811   11613  177631      DEC  -103
1812   11614  025754      LDB  AR2E   LOAD THE DIVIDED WORD
1813   11615  074073      SBP  *+1,C  AND COEAR THE SIGN BIT
1814   11616  065274      JMP  STMAX
1815*
1816   11617  021755      GOOD  LDA  AR2E+1
1817   11620  072110      RZA  *+2
1818   11621  065254      JMP  ZONK
1819*
1820   11622  170400      CMY
1821   11623  020132      LDA  N1    COMPLEMENT THE DIVISOR
1822   11624  024144      LDB  M13   INITIAL FIRST DIVIDED DIGIT TO -1
1823   11625  035477      STB  TEMP4  INITIALIZE THE DIGIT
1824*                                COUNT TO 13
1825   11626  024135      NWWD  LDB  M4    SET THE WORD DIGIT
1826   11627  035701      STB  TEMP4  COUNT TO 4
1827*
1828   11630  074604      NXDG  SBL  4    CLEAR LOCATION FOR NEXT
1829   11631  031476      STA  TEMP3  CALCULATED DIGIT AND ALLIGN
1830*                                STORE FIRST DIVIDED DIGIT
1831   11632  170420      DINC  FDV      SUBTRACT DIVISOR FROM DIVIDEND
1832   11633  074070      SIB  *+1    UNTIL OVERFLOW
1833   11634  045476      ISZ  TEMP3  INCREMENT THIS COUNT
1834   11635  067632      JMP  DINC   INCREMENT FIRST DIVIDED DIGIT
1835*                                JUMP BACK IF NOT ZERO
1836   11636  021755      LDA  AR2E+1  JUMP OUT OF
1837   11637  001756      ADA  AR2E+2  MAIN LOOP
1838   11640  001757      ADA  AR2E+3  IF DIVIDEND
1839   11641  072150      RZA  NO     IS ZERO (I.E. AN
1840   11642  070134      SES  NO,C   EXACT QUOTIENT

```

PAGE 209

MATH ROUTINED FOR 9830, CALLED 'MATH'

| | | | | | |
|-------|-------|--------|-------|-------------|-----------------------------------|
| 1841 | 11643 | 067663 | | JMP YES | WAS FOUND) |
| 1842* | | | | | |
| 1843 | 11644 | 170400 | NO | CMY | OTHERWISE RESTORE THE |
| 1844 | 11645 | 170560 | | FXA | DIVIDEND TO ITS LAST POSITIVE |
| | | | | | VALUE |
| 1845 | 11646 | 004132 | | ADB N1 | SUBTRACT ONE FROM THE LAST |
| | | | | | CALCULATED DIGIT |
| 1846 | 11647 | 170400 | | CMY | COMPLEMENT THE DIVIDEND AGAIN |
| 1847* | | | | | |
| 1848 | 11650 | 171400 | | MLS | SHIFT THE DIVIDEND LEFT ONE DIGIT |
| 1849 | 11651 | 000141 | | ADA N10 | LET THE NEW FIRST DIGIT BE THE |
| | | | | | ONE SHIFTED OUT-10 |
| 1850* | | | | | |
| 1851 | 11652 | 045477 | | ISZ TEMP4 | INCREMENT THE DIGIT COUNT |
| 1852 | 11653 | 070112 | | SAM *+2 | JUMP OUT OF LOOP IF DONE OR |
| 1853 | 11654 | 067665 | | JMP DONE | IF DIVISOR HAS A NON BCD DIGIT |
| 1854 | 11655 | 045701 | | ISZ TEMP4 | INCREMENT WORD DIGIT COUNT |
| 1855 | 11656 | 067630 | | JMP NXDG | JUMP BACK THRU LOOP IF NON ZERO |
| 1856 | 11657 | 135475 | | STB TEMP2,I | GLSE STORE LAST 4 DIGITS IN |
| | | | | | QUOTIENT |
| 1857 | 11660 | 045475 | | ISZ TEMP2 | INCREMENT THE QUOTIENT WORD |
| | | | | | POINTER |
| 1858 | 11661 | 067626 | | JMP NWWD | AND JUMP BACK |
| 1859* | | | | | |
| 1860 | 11662 | 074604 | | SDL 4 | ALIGN THE LAST DIGIT |
| 1861 | 11663 | 045701 | YES | ISZ TEMP4 | IF A PERFECT QUOTIENT |
| 1862 | 11664 | 067662 | | JMP *-2 | WAS FOUND |
| 1863* | | | | | |
| 1864 | 11665 | 135475 | DONE | STB TEMP2,I | STORE THE LAST 4 DIGITS |
| 1865 | 11666 | 020057 | | LDA ATEMP | TRANSFER |
| 1866 | 11667 | 000132 | | ADA N1 | THE QUOTIENT |
| 1867 | 11670 | 024214 | | LDB AR2A | INTO A |
| 1868 | 11671 | 170004 | | XFR | WORKING REGISTOR |
| 1869* | | | | | |
| 1870 | 11672 | 171450 | | NRM | NORMALIZE THE ANSWER |
| 1871 | 11673 | 074310 | | SZB .GOON | SKIP IF IT WAS NORMALIZED (B=0) |
| 1872* | | | | | |
| 1873 | 11674 | 021753 | | LDA AR2E-1 | LOAD LAST DIGIT |
| 1874 | 11675 | 050032 | | AND P15 | AND STORE IT |
| 1875 | 11676 | 001757 | | ADA AR2E+3 | IN THE |
| 1876 | 11677 | 031757 | | STA AR2E+3 | QUOTIENT |
| 1877 | 11700 | 024157 | | LDB N256 | LOAD AN EXP OF -1 INTO B |
| 1878* | | | | | |
| 1879 | 11701 | 105703 | .GOON | ADB ADR2,I | |
| 1880 | 11702 | 021744 | | LDA AR1E | |
| 1881 | 11703 | 070076 | | TCA | EXPONENT OF |
| 1882 | 11704 | 000017 | | ADA P2 | THE DIVISOR AND PASS INTO |
| 1883 | 11705 | 031744 | | STA AR1E | THE STORE ROUTINE |
| 1884 | 11706 | 065237 | | JMP STAR2 | |
| 1885* | | | | | |

PAGE 210

MATH ROUTINED FOR 9830, CALLED 'MATH'

```

1887*
1888*****
1889*
1890 11707 ORG 11707B
1891 11707 000000 BSS 11
1892 11722 001077 ABS ACS-*
1893 11723 001126 ABS ASN-*
1894*****
1895*
1896* LINKAGE
1897*
1898 11724 002010 ABS LOG-*
1899 11725 001330 ABS ATN-*
1900 11726 001125 ABS TAN-*
1901 11727 001060 ABS COS-*
1902 11730 001031 ABS SIN-*
1903 11731 005123 ABS SGN-*
1904 11732 004046 ABS INT-*
1905 11733 004227 ABS FSR-*
1906 11734 005155 ABS ABS-*
1907 11735 001407 ABS LNX-*
1908 11736 001333 ABS EXP-*
1909*
1910*

```

RESERVED FOR OTHER POINTERS

LOG BASE 10
ARC TANGENT
TANGENT
COSINE
SINE
SIGN
INTEGER VALUE
SQUARE ROOT
ABSOLUTE VALUE
NATURAL LOG
E TO THE POWER

PAGE 211 NUMBER CONVERSION ROUTINES

```

1912 17000          ORG 17000B
1913*
1914 01256          CLRA2   EQU CLAR2
1915 00051          LDVSR   EQU .46      POINTER TO .1E4
1916 01471          DIVSR   EQU L2T2
1917 17000 037400   B37.4   OCT 37400
1918*
1919* CONVERT FULL-PRECISION FLOATING NUMBER TO INTEGER
1920*
1921* FLOATING NUMBER IS IN AR2
1922*
1923*   JSM FLTFX
1924*   --- ERROR EXIT (OVERFLOW)
1925*   --- NORMAL RETURN WITH RESULT IN (B)
1926*
1927* NEGATIVE INTEGER IN 2'S COMPLEMENT FORM
1928*
1929 17001 074742   FLTFX   SBR 16      CLEAR B
1930 17002 021754   LDA AR2   LOOK AT EXPONENT
1931 17003 070652   SAM PP2   DONE IF FRACTIONAL
1932 17004 000157   FLT1     ADA M256   ELSE DECREMENT 10'S EXPONENT
1933 17005 031471   STA L2T2   AND SAVE IT.
1934 17006 020061   LDA .10    *10
1935 17007 061201   JSM IMPY
1936*
1937* ON RETURN THE A-REG IS SET NEGATIVE ON OVERFLOW, TESTING
1938* IS NOT MADE UNTIL THE NEXT DIGIT IS ADDED IN.
1939*
1940 17010 070137   LDB A
1941 17011 171400   MLS
1942 17012 070037   ADB A      NEXT DIGIT
1943 17013 074352   SBM RET1   ADD NEW DIGIT
1944 17014 021471   LDA L2T2   OVERFLOW?
1945 17015 071353   SAP FLT1   RECALL EXPONENT
1946 17016 070111   SLA *+2    LOOP IF POSITIVE
1947 17017 074076   TCB        TEST MANTISSA SIGN
1948 17020 064357   PP2      JMP RETN2
1949*
1950* CONVERT FULL-PRECISION FLOATING NUMBER TO ROUNDED INTEGER
1951*
1952* FLOATING NUMBER IN AR2
1953*   JSM FLTRF
1954*   --- ERROR EXIT
1955*   --- NORMAL RETURN WITH RESULT IN B
1956*
1957 17021 063001   FLTRF   JSM FLTFX   GO GET INTEGER
1958 17022 064444   RET1    JMP XFAR2   P+1 RETURN FOR OVERFLOW ERRORS
1959 17023 076210   STA L2T2   RZB *+4   IS (B)=0?
1960 17024 031471   ADA B400   YES, SAVE AR2 EXP AT L2T2
1961 17025 000013   SAM PP2   ADD EXP OF 1 TO CHECK IF <.1
1962 17026 071512   MLS      IF EXP IS STILL NEGATIVE RETURN
1963 17027 171400   ADA M5    GET MS FRACTIONAL DIGIT
1964 17030 000136   SAM PP2   ADD -5
1965 17031 071352   SBP *+2   DONE IF .LT. 5
1966 17032 074113   TCB      IS B POSITIVE?
1967 17033 074076   NO — MAKE IT SO

```

328

316

1,444,141

316

PAGE 212

NUMBER CONVERSION ROUTINES

```

1968 17034 074070 SIB *+1 ADD 1 FOR ROUNDING
1969 17035 067013 JMP FLT2 USE FLT2 TO RESTORE SIGN,
1970* AND OVERFLOW TEST.
1971*
1972*
1973*
1974* OUTPUT AN INTEGER GIVEN IN (B) TO THE I/O BUFFER
1975*
1976 17036 063055 OUTIN JSM FXFLT
1977 17037 035470 STB L2T1 (B)=EXPONENT WORD OF AR2
1978 17040 061314 JSM ODGIT
1979 17041 025470 LDB L2T1 LOOK AT EXPONENT
1980 17042 004157 ADB M256 DECREMENT 10'S EXPONENT
1981 17043 075613 SBP *-4 LOOP IF POSITIVE
1982 17044 170402 .R1 RET
1983*
1984* TRANSFER A BCD DIGIT INTO D12 OF AR2
1985*
1986 17045 031470 STA L2T1 ALTERNATE ENTRY POINT
1987 17046 171400 XFER. MLS
1988 17047 021757 LDA AR2+3
1989 17050 041470 IOR L2T1
1990 17051 031757 STA AR2+3
1991 17052 170402 RET
1992*
1993*****
1994*
1995 17053 000000 BSS 1 CHECKSUM WORD FOR THIS 256 WORD
1996* BLOCK
1997*****

```

PAGE 213

NUMBER CONVERSION ROUTINES

1999*

SIGN ROUTINE

2000*

2001*

ON RETURN AR2 = 1 IF ARG WAS +, 0 IF 0, AND -1 IF NEGATIVE

2002*

2003*

2004

17054 061317 SGN JSM SGNS SGN FLOWS INTO FXFLT

2005*

2006*

CONVERT INTEGER TO FULL-PRECISION FLOATING NUMBER

2007*

FLOATING NUMBER BUILT IN AR2

2008*

2009*

LDB N

2010*

JSM FXFLT

2011*

2012

17055 061256 FXFLT JSM CLRA2

CLEAR AR2

2013

17056 075310 SZB .R1

RETURN IF A IS ZERO

2014

17057 074153 SBP *+3

AND GET ABSOLUTE VALUE

2015

17060 074076 TCB

MAKE B POSITIVE

2016

17061 045754 ISZ AR2

NOTE THE SIGN OF THE INTEGER

2017

17062 020051 LDA LDVSR

INITIALIZE THE DIVISOR POINTER

2018

17063 031471 STA DIVSR

2019

17064 121471 FXFL LDA DIVSR,I

2020

17065 070513 SAP .IK

2021

17066 035470 STB L2T1

2022

17067 063046 JSM XFER

UNITS POSITION

2023

17070 171450 NRM

2024

17071 004142 ADB M11

COMPUTE EXPONENT

2025

17072 074076 TCB

PLACEMENT

2026

17073 074404 SBL 8

SHIFT B LEFT 8

2027

17074 005754 ADB AR2

MERGE IN THE SIGN BIT

2028

17075 035754 STB AR2

STORE IT IN AR2

2029

17076 170402 RET

2030

17077 070076 .IK TCA

2031

17100 031467 STA LOCL2

STORE NEGATIVE DIVISOR

2032

17101 020132 LDA M1

2033

17102 070070 SIA *+1

INCREMENT INTEGER COUNTER

2034

17103 005467 ADB LOCL2

2035

17104 075713 SBP *-2

2036

17105 105471 ADB DIVSR,I

RESTORE DIVIDEND

2037

17106 063045 JSM XFER-1

MERGE A INTO D12 SLOT

2038

17107 045471 ISZ DIVSR

ADVANCE TO NEXT DIVISOR

IN TABLE

2039

17110 067064 JMP FXFL

2040*****

2041*

ABSOLUTE VALUE

2042*

2043

17111 060445 ABS JSM XFAR2+1

2044

17112 025754 LDB AR2

2045

17113 074071 SLB *+1,C

2046

17114 067075 JMP .IK-2

2047*

2048*****

2049*

2050*

CONVERT FULL-PRECISION FLOATING NUMBER TO SPLIT

2051*

2052*

FLOATING NUMBER IS IN AR2

2053*

LDB ADDRESS FOR SPLIT

2054*

JSM FLTSP

380

318

1,444,141

318

PAGE 214

NUMBER CONVERSION ROUTINES

| | | | | | | |
|-------|-------|--------|-------|-----|---------|-----------------------------------|
| 2055* | | | | | | |
| 2056 | 17115 | 160205 | SPLUF | JSM | AERRA,I | THE EXPONENT=10XXXXXX |
| 2057 | 17116 | 177625 | | DEC | -107 | THIS IS AN UNDERFLOW CONDITION |
| 2058 | 17117 | 061256 | | JSM | CLRA2 | CONDITION SO CLEAR AR2 |
| 2059 | 17120 | 067122 | | JMP | FLTSP+1 | AR2 IS ZERO — PROCEED |
| 2060* | | | | | | |
| 2061 | 17121 | 035727 | FLTSP | STB | MBIN2 | SAVE STARTING ADD. OF SPLIT |
| 2062 | 17122 | 021754 | | LDA | AR2 | GET SIGN |
| 2063 | 17123 | 050012 | | AND | .1 | OF MANTISSA |
| 2064 | 17124 | 031726 | | STA | MBIN1 | AND SAVE |
| 2065 | 17125 | 025754 | | LDB | AR2 | LOAD EXPONENT FOR OVERFLOW |
| | | | | | | CHECK |
| 2066 | 17126 | 074706 | | RBR | 15 | POSITION BIT 15 IN LSB |
| 2067 | 17127 | 074111 | | SLB | *+2 | IS BIT 15=1? |
| 2068 | 17130 | 075273 | | SBP | SPLUF,C | YES — IF BIT 14=1; CONTINUE |
| 2069 | 17131 | 074333 | | SBP | OK1,C | NO — IF BIT 14=0; JUMP OVER ERROR |
| 2070* | | | | | | |
| 2071 | 17132 | 160205 | | JSM | AERRA,I | OVERFLOW ERROR |
| 2072 | 17133 | 177626 | | DEC | -106 | |
| 2073 | 17134 | 061274 | | JSM | STMAX | SET AR2=THE MAX NUMBER |
| 2074 | 17135 | 023000 | | LDA | B37.4 | SET EXPONENT |
| 2075 | 17136 | 031754 | | STA | AR2 | =E63 |
| 2076* | | | | | | |
| 2077 | 17137 | 024017 | OK1 | LDB | .2 | |
| 2078 | 17140 | 070742 | | SAR | 16 | |
| 2079 | 17141 | 174470 | | MRY | | SHIFT AR2 MANTISSA RIGHT |
| 2080 | 17142 | 021754 | | LDA | AR2 | PICK UP EXPONENT WORD |
| 2081 | 17143 | 070300 | | AAR | 7 | SHIFT OFF 7 BITS |
| 2082 | 17144 | 041726 | | IOR | MBIN1 | ADD IN MANTISSA SIGN |
| 2083 | 17145 | 070404 | | SAL | 8 | |
| 2084 | 17146 | 041755 | | IOR | AR2+1 | ADD IN FIRST TWO DIGITS |
| 2085 | 17147 | 131727 | | STA | MBIN2,I | STASH FIRST WORD |
| 2086 | 17150 | 045727 | | ISZ | MBIN2 | |
| 2087 | 17151 | 021756 | | LDA | AR2+2 | |
| 2088 | 17152 | 131727 | | STA | MBIN2,I | STASH SECOND WORD |
| 2089 | 17153 | 170402 | | RET | | |
| 2090* | | | | | | |

| Address | Instruction | Operation | Comments |
|---------|---------------------------------|-----------|--|
| 2092* | BUILD A NUMBER IN AR2 | | |
| 2093* | FIRST CHARACTER IN A | | |
| 2094* | | | |
| 2095* | | | |
| 2096 | 17154 070137 NUMCK LDB A | | |
| 2097 | 17155 061256 JSM CLRA2 | | CLEAR AR2 |
| 2098 | 17156 020132 LDA M1 | | |
| 2099 | 17157 031745 STA EXCNT | | SET EXPONENT COUNT TO -1 |
| 2100 | 17160 031746 STA DPFLG | | SET DECIMAL POINT FLAG TO -1 |
| 2101 | 17161 020143 LDA M12 | | |
| 2102 | 17162 031744 STA SIGDG | | SET SIGNIFICANT DIGIT COUNT TO -12 |
| 2103 | 17163 074117 LDA B | | RECALL CHARACTER |
| 2104 | 17164 067175 JMP NUMC1 | | |
| 2105* | | | |
| 2106* | PROCESS A DECIMAL POINT | | |
| 2107* | | | |
| 2108 | 17165 021746 .ANUM LDA DPFLG | | DECIMAL POINT YET? |
| 2109 | 17166 072250 RZA NUMC1-2 | | NO — SET NUMBER FOUND AND GET NEXT CHAR. |
| 2110 | 17167 055745 DSZ EXCNT | | YES — DECR. EXPONENT COUNT |
| 2111 | 17170 067174 JMP NUMC1-1 | | |
| 2112 | 17171 045746 NUMCP ISZ DPFLG | | SET FLAG |
| 2113 | 17172 160206 JSM ERRA,1 | | FLAG ALREADY SET, 2 DECIMAL POINTS |
| 2114 | 17173 045754 ISZ AR2 | | SET EXPONENT NON-ZERO; NUMBER FOUND |
| 2115 | 17174 061055 JSM GNEXT | | GET NEXT CHAR. |
| 2116 | 17175 060750 NUMC1 JSM DIGCK | | DIGIT? |
| 2117 | 17176 067215 JMP NUMC2 | | NO — IT'S NOT A DIGIT |
| 2118 | 17177 076210 RZB NUMCD | | ZERO? |
| 2119 | 17200 021744 LDA SIGDG | | YES, |
| 2120 | 17201 010143 CPA M12 | | ANY SIGNIFICANT DIGITS YET? |
| 2121 | 17202 067165 JMP .ANUM | | NO — NON-SIGNIFICANT ZERO |
| 2122* | PROCESS A SIGNIFICANT DIGIT | | |
| 2123 | 17203 021746 NUMCD LDA DPFLG | | DECIMAL POINT FOUND YET? |
| 2124 | 17204 070150 SZB *+3 | | |
| 2125 | 17205 045745 ISZ EXCNT | | NO, INC COUNT |
| 2126 | 17206 067207 JMP *+1 | | |
| 2127 | 17207 021744 LDA SIGDG | | 12 DIGITS |
| 2128 | 17210 070230 SIA NUML1 | | FOUND YET? IF YES SKIP |
| 2129 | 17211 031744 STA SIGDG | | UPDATE COUNT |
| 2130 | 17212 035470 STB L2T1 | | MERGE LATEST DIGIT |
| 2131 | 17213 063046 JSM XFER | | INTO D12 OF AR2 |
| 2132 | 17214 067173 NUML1 JMP NUMC1-2 | | |
| 2133* | | | |
| 2134* | PROCESS NON-DIGIT MANTISSA CHAR | | |
| 2135* | | | |
| 2136* | | | |
| 2137 | 17215 010051 NUMC2 CPA .46 | | DECIMAL POINT? |
| 2138 | 17216 067171 JMP NUMCP | | YES |
| 2139 | 17217 025754 LDB AR2 | | |
| 2140 | 17220 076110 RZB *+2 | | ANY MANTISSA FOUND? |
| 2141 | 17221 170402 RET | | NO, RETURN TO P+1 |
| 2142 | 17222 010071 CPA E | | "E"? |
| 2143 | 17223 067250 JMP NUMCE | | YES |
| 2144 | 17224 074742 SBR 16 | | NO, SET 0 EXPONENT |
| 2145 | 17225 072053 SAP *+1,S | | SET FIXED POINT FLAG |
| 2146* | CHARACTER IS NOT PART OF NUMBER | | |
| 2147 | 17226 045746 NUMC3 ISZ DPFLG | | COMPLEMENT EXPONENT |

PAGE 216

NUMBER CONVERSION ROUTINES

| | | | | | |
|-------|------------------|--------|-----|--------|---|
| 2148 | 17227 | 067231 | JMP | *+2 | |
| 2149 | 17230 | 074076 | TCB | | |
| 2150 | 17231 | 005745 | ADB | EXCNT | IF NEGATIVE
ADD EXPONENT DUE TO DECIMAL
PLACEMENT |
| 2151 | 17232 | 035745 | STB | EXCNT | STORE EXPONENT |
| 2152 | 17233 | 171450 | NRM | | LEFT JUSTIFY MANTISSA |
| 2153 | 17234 | 025745 | LDB | EXCNT | RECALL EXPONENT |
| 2154 | 17235 | 074404 | SBL | 8 | LEFT JUSTIFY EXPONENT |
| 2155 | 17236 | 045721 | ISZ | .SIGN | .SIGN WAS SET IN "CONST" ROUTINE. |
| 2156 | 17237 | 067241 | JMP | *+2 | |
| 2157 | 17240 | 074070 | SIB | *+1 | ADD IN SIGN OF MANTISSA |
| 2158 | 17241 | 070115 | SEC | *+2 | IF MANTISSA IS 0 |
| 2159 | 17242 | 074742 | SBR | 16 | THEN SET EXPONENT TO 0 |
| 2160 | 17243 | 035754 | STB | AR2 | STORE EXPONENT WORD |
| 2161 | 17244 | 004021 | ADB | .4 | SET FLOATING FLAG (BIT 2) |
| 2162 | 17245 | 070133 | SAP | *2+2,C | LEAVE (B) AT .4 IF THERE WAS AN
EXPONENT |
| 2163 | 17246 | 004133 | ADB | M2 | NO, SET FIXED FLAG (BIT 1) |
| 2164 | 17247 | 064357 | JMP | RETN2 | RETURN TO P+2 |
| 2165* | PROCESS EXPONENT | | | | |
| 2166 | 17250 | 024132 | LDB | M1 | |
| 2167 | 17251 | 035724 | STB | GFLAG | SET INTCK TO RETURN ON ERROR |
| 2168 | 17252 | 061166 | JSM | GSIGN | GET EXPONENT SIGN |
| 2169 | 17253 | 035746 | STB | DPFLG | SAVE IT |
| 2170 | 17254 | 060750 | JSM | DIGCK | DIGIT? |
| 2171 | 17255 | 160206 | JSM | ERRA,I | NO, MISSING EXPONENT |
| 2172 | 17256 | 025745 | LDB | EXCNT | COMPUTE — (ABSOLUTE VALUE |
| 2173 | 17257 | 004156 | ADB | M100 | OF MAX. ALLOWABLE EXPONENT) |
| 2174 | 17260 | 060676 | JSM | INTCK | GET ABSOLUTE VALUE OF EXPONENT |
| 2175 | 17261 | 067264 | JMP | *+3 | |
| 2176 | 17262 | 077550 | RZB | NUMBER | EXPONENT OVERFLOW |
| 2177 | 17263 | 067226 | JMP | NUMC3 | EXPONENT OF ZERO GIVEN |
| 2178 | 17264 | 055706 | DSZ | \$BPTR | RESET BUFFER POINTER |
| 2179 | 17265 | 067226 | JMP | NUMC3 | EXPONENT OF ZERO GIVEN |
| 2180 | 01744 | | EQU | AR1 | |
| 2181 | 01745 | | EQU | AR1+1 | |
| 2182 | 01746 | | EQU | AR1+2 | |

PAGE 217

NUMBER CONVERSION ROUTINES

2184*
 2185* CONVERT A NUMBER IN AR2 INTO SYNTAX FORMAT AND STORE IT
 2186* (B)=EXPONENT WORD
 2187* (A)=NEXT CHARACTER, MUST BE SAVED.
 2188*
 2189 17266 031474 NUMOP STA TEMP1
 2190 17267 074117 LDA B
 2191 17270 050105 AND B377 GET MANTISSA SIGN AND FLOAT
 2192 17271 070006 RAR 1 FLAG
 2193 17272 072113 SET MANTISSA SIGN IN BIT 9
 2194 17273 040107
 2195 17274 024030 LDB .12
 2196 17275 005744 ADB SIGDG GET SIGNIFICANT DIGIT COUNT
 2197 17276 076110 RZB *+2
 2198 17277 074070 SIB *+1 IF 0, SET TO 1
 2199 17300 074544 SBL 5 MOVE TO OPERAND FIELD
 2200 17301 074217 IOR B MERGE
 2201 17302 055706 DSZ SBPTR
 2202 17303 141706 IOR SBPTR,I COMBINE WITH OPERAND
 2203 17304 131706 STA SBPTR,I AND STORE
 2204 17305 074202 NUMO1 SBR 5 COMPUTE NUMBER OF
 2205 17306 004022 ADB .5
 2206 17307 074042 SBR 2 WORDS TO BE TRANSFERRED +1
 2207 17310 035477 STB TEMP4 SAVE COUNT
 2208 17311 021754 LDA AR2
 2209 17312 050157 AND M256 GET EXPONENT
 2210 17313 025755 LDB AR2+1
 2211 17314 074342 SBR 8
 2212 17315 074217 IOR B COMBINE WITH D1 AND D2
 2213 17316 060456 JSM SBPUD UPDATE POINTER
 2214 17317 171400 MLS
 2215 17320 171400 MLS
 2216 17321 024214 LDB AR2A SHIFT MANTISSA
 2217 17322 074070 NUMO2 SIB *+1 GET MANTISSA ADDRESS
 2218 17323 074517 LDA B,I UPDATE MANTISSA ADDRESS
 2219 17324 060456 JSM SBPUD GET READY TO STORE WORD IN SYN.
 2220 17325 055477 DSZ TEMP4 BUFF.
 2221 17326 067322 JMP NUMO2 UPDATE SYNTAX ADDRESS
 2222 17327 021474 LDA TEMP1 MORE WORDS TO BE
 2223 17330 170402 RET TRANSFERRED?
 YES
 NO, RECALL CHAR AND RETURN

334

322

1,444,141

322

PAGE 218

NUMBER CONVERSION ROUTINES

2225* SNFLT ROUTINE

2226*

2227* THIS ROUTINE TRANSFERS A
 2228* NUMBER IN SYNTAX FORMAT TO
 2229* AR2 AND STORES IT IN STANDARD
 2230* FORMAT

2231*

2232* ENTRY CONDITIONS:

2233*

2234* TEMPS=WORD POINTER TO
 2235* START OF NUMBER

2236*

2237* EXIT CONDITIONS:

2238*

2239* TEMPS=WORD POINTER TO FIRST
 2240* WORD FOLLOWING NUMBER
 2241* AR2=NUMBER IN STANDARD FORM

2242*

2243*

| | | | | | | |
|-------|-------|--------|-------|-----|---------|---|
| 2244 | 17331 | 021472 | SNFLT | LDA | TEMPS | |
| 2245 | 17332 | 070070 | | SIA | *+1 | TRANSFER THE NUMBER |
| 2246 | 17333 | 024214 | | LDB | AR2A | INTO AR2 |
| 2247 | 17334 | 170004 | | XFR | | |
| 2248* | | | | | | |
| 2249 | 17335 | 121472 | | LDA | TEMPS,I | |
| 2250 | 17336 | 031470 | | STA | L2T1 | |
| 2251 | 17337 | 070202 | | SAR | 5 | |
| 2252 | 17340 | 050032 | | AND | B17 | GET THE # OF DIGITS |
| 2253 | 17341 | 000022 | | ADA | .5 | |
| 2254 | 17342 | 070042 | | SAR | 2 | COMPUTE HOW MANY WORDS THIS IS |
| 2255 | 17343 | 070137 | | LDB | A | |
| 2256 | 17344 | 001472 | | ADA | TEMPS | |
| 2257 | 17345 | 031472 | | STA | TEMPS | MOVE TEMPS TO THE LAST WORD OF
THE NUMBER |
| 2258 | 17346 | 004214 | | ADB | AR2A | |
| 2259 | 17347 | 070742 | | SAR | 16 | |
| 2260 | 17350 | 014112 | | CPB | B1760 | |
| 2261 | 17351 | 067355 | | JMP | *+4 | |
| 2262 | 17352 | 074557 | | STA | B,I | CLEAR THE REMAINDER OF
THE MANTISSA DIGITS |
| 2263 | 17353 | 074070 | | SIB | *+1 | |
| 2264 | 17354 | 067350 | | JMP | *-4 | |
| 2265* | | | | | | |
| 2266 | 17355 | 024017 | | LDB | 2 | RIGHT SHIFT AR2
TWO PLACES |
| 2267 | 17356 | 174470 | | MRY | | |
| 2268 | 17357 | 021754 | | LDA | AR2 | |
| 2269 | 17360 | 070404 | | SAL | 8 | |
| 2270 | 17361 | 041755 | | IOR | AR2+1 | MERGE INTO AR2+1 DIGITS ONE AND
TWO |
| 2271 | 17362 | 031755 | | STA | AR2+1 | |
| 2272 | 17363 | 021754 | | LDA | AR2 | LOAD THE EXPONENT
AND MASK |
| 2273 | 17364 | 050157 | | AND | M256 | |
| 2274 | 17365 | 025470 | | LDB | L2T1 | |
| 2275 | 17366 | 074402 | | SBR | 9 | INCLUDE THE SIGN BIT |
| 2276 | 17367 | 074111 | | SLB | *+2 | |
| 2277 | 17370 | 070070 | | SIA | *+1 | |
| 2278 | 17371 | 031754 | | STA | AR2 | |
| 2279 | 17372 | 170402 | | RET | | |
| 2280 | ***** | | | | | |

PAGE 219

MORE 9830 FUNCTIONS

2282*****

2283 16000 ORG 16000B

2284*

2285*

2286* INTEGER ROUTINE

2287*

2288 16000 062015 INT JSM FDFRC TRANSFER ARG. INTO AR1, AND TEST
FOR FRACTIONAL

2289 16001 066005 JMP *+4 P+1, FRACTIONAL ARGUMENT

2290*

2291 16002 060444 JSM XFAR2 P+2, NON FRACTIONAL ARG.

2292 16003 171450 NRM TRANSFER TO AR2, AND NORMALIZE

2293 16004 065254 JMP ZONK

2294*

2295 16005 062002 JSM INT+2

2296 16006 025744 LDB AR1 LOAD (B) WITH THE EXPONENT

2297 16007 074113 SBP *+2 IF THE EXPONENT WAS NEGATIVE

2298 16010 061256 JSM CLAR2 CLEAR AR2

2299 16011 075551 SLB INT+4 RETURN IF THE ARG WAS +.

2300 16012 020214 LDA AR2A THE ARGUMENT WAS - AND IT HAD A

2301 16013 024020 LDB ONE FRACTIONAL PART

2302 16014 164217 JMP .FSBA,I SO SUBTRACT ONE FROM IT

2303*

2304*****

2305*

2306*

2307* SUBROUTINE FOR INT AND UPARROW

2308*

2309 16015 024213 FDFRC LDB AR1A

2310 16016 170004 XFR

2311 16017 020012 LDA .1 SET THE FLAG

2312 16020 031714 STA SAVE

2313 16021 025744 LDB AR1 LOAD THE EXPONENT

2314 16022 074512 SBM UP1+7 EXIT IF EXP IS FRACTIONAL

2315 16023 074340 ABR 8

2316 16024 004142 ADB M11

2317*

2318 16025 074313 UP1 SBP *+6 MUST BE AN INTEGER IF (B) IS +

2319 16026 170410 DRS RIGHT SHIFT AR1

2320 16027 074070 SIB *+1

2321 16030 071650 SZA UP1 SKIP BACK IS SHIFTED DIGIT WAS ZERO

2322 16031 045714 ISZ SAVE OTHERWISE INCR. THE FLAG FOR

FRACTIONAL PART

2323 16032 066025 JMP UP1 SKIP BACK TO FINISH INTEGERATION

2324 16033 055714 DSZ SAVE SKIP IF THERE WAS NO FRACTIONAL

PART

2325 16034 170402 RET P+1 RETURN FOR FRACTIONAL PART

2326 16035 064357 JMP RETN2 P+2 IF IT WAS AN INTEGER

PAGE 220

MORE 9830 FUNCTIONS

2328*
 2329 16036 001734 ADRYY OCT 1734 ADDRESS OF REGISTOR Y
 2330 16037 013344 ILNX DEF LNX
 2331 16040 012550 IINIT DEF INIT+1
 2332 16041 013266 I.EXP DEF EXP-3
 2333*
 2334* RET4 IS USED WHEN THE NUMBER IS ZERO
 2335*
 2336 16042 021741 RET4 LDA XC+1 LOAD DIGITS 1-4 OF THE POWER
 2337 16043 070410 SZA ER49 IF THE POWER IS ZERO ERROR 49
 2338 16044 021740 LDA XC LOAD THE EXPONENT OF THE POWER
 2339 16045 074742 SBR 16 CLEAR (B) FOR STMAX ROUTINE
 2340 16046 070211 SLA *+4 RETURN IF THE POWER IS POSITIVE;
 2341 16047 061274 JSM STMAX SET AR2 = +9.9999E99: NEGATIVE POWER AR2=0
 2342 16050 160205 JSM AERRA,I
 2343 16051 177630 DEC -104
 2344 16052 170402 RET
 2345*
 2346 16053 160206 ER49 JSM ERRA,I 010 ERROR
 2347*
 2348*****
 2349*
 2350* RAISE A NUMBER TO A POWER
 2351*
 2352*****
 2353*
 2354 16054 035702 UPARO STB ADR1 STORE THE ADDRESS OF THE POWER,
 (A) IS SAVED BY INIT
 2355*
 2356 16055 162040 JSM IINIT,I
 2357 16056 021702 LDA ADR1 TRANSFER THE POWER INTO
 2358 16057 024111 LDB B1740 XC REGISTER
 2359 16060 170004 XFR
 2360 16061 071055 SEC RET4 STORE ZERO IF IT IS ZERO
 2361*
 2362 16062 021702 LDA ADR1 LOAD THE ADDRESS OF THE POWER
 2363 16063 362015 JSM FDFRC GO CHECK IF THERE IS A FRACTIONAL
 PART
 2364 16064 066137 JMP .UPA1 YES THERE WAS
 2365*
 2366 16065 025747 UP3 LDB AR1+3 SAVE THE D12 DIGIT IN CASE THERE
 2367 16066 035714 STB SAVE IS AN OVERFLOW IN FLTFA
 2368 16067 021702 LDA ADR1
 2369 16070 060445 JSM XFAR2+1
 2370 16071 160226 JSM FLTFA,I CONVERT AR2 TO AN INTEGER
 2371 16072 066153 JMP UP4 IT'S AN INTEGER BUT TO LARGE
 2372*
 2373* RAISE AN ARGUMENT TO AN INTEGER POWER
 2374*
 2375 16073 035714 IPWR STB SAVE
 2376 16074 074113 SBP *+2
 2377 16075 074076 TCB
 2378 16076 035710 STB COUNT SAVE THE ABS(POWER)
 2379 16077 020020 LDA ONE
 2380 16100 026036 LDB ADRYY INITIALIZE THE RESULT TO ONE
 2381 16101 170004 XFR
 2382 16102 021703 LDA ADR2
 2383 16103 024057 LDB ATEMP PLACE THE BASE IN REGISTOR Y

PAGE 221

MORE 9830 FUNCTIONS

| | | | | | |
|-------|-------|--------|------|-------------|---|
| 2384 | 16104 | 170004 | | XFR | |
| 2385* | | | | | |
| 2386 | 16105 | 021710 | IPW1 | LDA COUNT | LOAD THE POWER |
| 2387 | 16106 | 070411 | | SLA IPW2 | IS THE POWER ODD? |
| 2388* | | | | | |
| 2389 | 16107 | 020057 | | LDA ATEMP | YES |
| 2390 | 16110 | 026036 | | LDB ADRYY | |
| 2391 | 16111 | 160220 | | JSM .FMPI,I | MULTIPLY BY PARTIAL RESULT |
| 2392 | 16112 | 026036 | | LDB ADRYY | |
| 2393 | 16113 | 020214 | | LDA AR2A | AND PLACE IT BACK |
| 2394 | 16114 | 170004 | | XFR | |
| 2395 | 16115 | 021710 | | LDA COUNT | |
| 2396* | | | | | |
| 2397 | 16116 | 070002 | IPW2 | SAR 1 | EVEN POWER ENTRY POINT, DIVIDE
POWER BY 2 |
| 2398 | 16117 | 031710 | | STA COUNT | AND RESTORE |
| 2399* | | | | | |
| 2400 | 16120 | 070410 | | SZA DPW | IF THE POWER IS ZERO, JUMP TO EXIT
PART |
| 2401* | | | | | |
| 2402 | 16121 | 020057 | | LDA ATEMP | |
| 2403 | 16122 | 024057 | | LDB ATEMP | |
| 2404 | 16123 | 160220 | | JSM .FMPI,I | THE POWER WAS EVEN SO MULTIPLY
IT BY ITSELF |
| 2405 | 16124 | 024057 | | LDB ATEMP | AND PLACE IT BACK |
| 2406 | 16125 | 020214 | | LDA AR2A | |
| 2407 | 16126 | 170004 | | XFR | |
| 2408 | 16127 | 066105 | | JMP IPW1 | |
| 2409* | | | | | |
| 2410 | 16130 | 021714 | DPW | LDA SAVE | ALMOST DONE |
| 2411 | 16131 | 070152 | | SAM *+3 | WAS THE SIGN OF THE EXPONENT NEG? |
| 2412 | 16132 | 022036 | | LDA ADRYY | NO, TRANSFER THE ANSWER INTO AR2 |
| 2413 | 16133 | 064445 | | JMP XFAR2+1 | |
| 2414 | 16134 | 020020 | | LDA ONE | YES, TAKE THE RECIPICAL OF THE
ANSWER |
| 2415 | 16135 | 026036 | | LDB ADRYY | |
| 2416 | 16136 | 164221 | | JMP .FDVA,I | |
| 2417* | | | | | |
| 2418* | | | | | |
| 2419* | | | | | |
| 2420* | | | | | |
| 2421 | 16137 | 125703 | UPA1 | LDB ADR2,I | LOAD THE EXPONENT WORD OF THE
BASE |
| 2422 | 16140 | 074131 | | SLB UP2,C | SKIP IF IT IS POSITIVE |
| 2423* | | | | | |
| 2424* | | | | | |
| 2425 | 16141 | 160206 | ER50 | JSM ERRA,I | (-BASE)↑ (FRACTIONAL EXPOENT) |
| 2426* | | | | | |
| 2427* | | | | | |
| 2428 | 16142 | 035714 | UP2 | STB SAVE | |
| 2429 | 16143 | 021703 | | LDA ADR2 | LOAD THE ADDRESS OF THE BASE |
| 2430 | 16144 | 162037 | | JSM ILNX,I | TAKE NATURAL LOG OF THE BASE |
| 2431 | 16145 | 024111 | | LDB B1740 | LOAD THE ADDRESS OF THE EXPONENT
(SAVED IN XC) |
| 2432 | 16146 | 162041 | | JSM I.EXP,I | EXP (POWER * LOG (BASE)) |
| 2433* | | | | | |
| 2434 | 16147 | 021714 | | LDA SAVE | LOAD THE LAST DIGIT OF THE ORIGINAL
EXPOENET |
| 2435 | 16150 | 070451 | | SLA OVER | |
| 2436 | 16151 | 045754 | | ISZ AR2 | NEGATE THE ANSWER IF IT IS ODD |
| 2437 | 16152 | 065254 | | JMP ZONK | STORE ANSWER |
| 2438* | | | | | |
| 2439* | | | | | |

338

326

1,444,141

326

PAGE 222

MORE 9830 FUNCTIONS

2440* INTEGER POWER, BUT GREATER THAN 32676

2441*

2442 16153 125703 UP4 LDB ADR2,I

2443 16154 074131 SLB *+2,C

2444 16155 066157 JMP *+2

2445 16156 035714 STB SAVE

2446 16157 135703 STB ADR2,I

2447 16160 066143 JMP UP2+1

2448*

THE BASE WAS +, SO CLEAR BIT 0
TAKE THE ABS(BASE)

PAGE 223

MORE 9830 FUNCTIONS

| | | | | | | |
|-------|-------|--------|----------------------|-------------|---------------------------------|---------|
| 2450* | | | | | | |
| 2451 | 16161 | 170402 | OVER | RET | | |
| 2452* | | | | | | |
| 2453* | | | FLOATING SQUARE ROOT | | | |
| 2454* | | | | | | |
| 2455* | | | | | | |
| 2456 | 16162 | 024213 | FSR | LDB AR1A | | |
| 2457 | 16163 | 170004 | | XFR | | |
| 2458 | 16164 | 020214 | | LDA AR2A | | |
| 2459 | 16165 | 170000 | | CLR | | |
| 2460* | | | | | | |
| 2461 | 16166 | 021745 | | LDA AR1E+1 | IF THE ARGUMENT | |
| 2462 | 16167 | 071510 | | SZA OVER | IS ZERO RETURN | |
| 2463* | | | | | | |
| 2464* | | | | | | |
| 2465 | 16170 | 024022 | | LDB P5 | MULTIPLY THE ARGUMENT | |
| 2466 | 16171 | 171460 | | FMP | BY 5 | |
| 2467 | 16172 | 025744 | | LDB AR1E | RIGHT SHIFT THE ARGUMENT | |
| 2468 | 16173 | 074444 | | SBL 7 | ONCE IF THE EXPONENT IS | |
| 2469 | 16174 | 074112 | | SBM *+2 | ODD AND TWICE IF ITS GIVEN | |
| 2470 | 16175 | 074070 | | SIB *+1 | AND ONCE MORE IF A CARRY | |
| 2471 | 16176 | 074070 | | SIB *+1 | EXISTS FROM THE MULTIPLY | |
| 2472* | | | | | | |
| 2473 | 16177 | 061306 | | JSM ROUND+1 | ROUND IF THE LAST SHIFTED DIGIT | IS >= 5 |
| 2474* | | | | | | |
| 2475 | 16200 | 020213 | | LDA AR1A | INITIALIZE THE ANSWER | |
| 2476 | 16201 | 070070 | | SIA *+1 | ANSWER WORD POINTER 1 AND 2 | |
| 2477 | 16202 | 031701 | | STA TEMP4 | TO THE SECOND WORD OF THE | |
| 2478 | 16203 | 031476 | | STA TEMP3 | WORKING REGISTER | |
| 2479 | 16204 | 170000 | | CLR | CLEAR THE WORKING REGISTER | |
| 2480* | | | | | | |
| 2481 | 16205 | 021744 | | LDA AR1E | IF THE RADICAND IS | |
| 2482 | 16206 | 070131 | | SLA *+2,C | NEGATIVE — RETURN. | |
| 2483 | 16207 | 160206 | ER48 | JSM ERRA,I | | |
| 2484* | | | | | | |
| 2485 | 16210 | 070000 | | AAR 1 | DIVIDE THE EXPONENT WORD BY 2, | |
| 2486 | 16211 | 050163 | | AND ZAP | CLEAR BITS 1—7 | |
| 2487 | 16212 | 031744 | | STA AR1E | AND STORE | |
| 2488* | | | | | | |
| 2489 | 16213 | 020117 | | LDA AD5 | INITIALIZE THE | |
| 2490 | 16214 | 031475 | | STA TEMP2 | 'FIVE' WORD TO 0500 | |
| 2491 | 16215 | 020004 | | LDA D1000 | INITIALIZE THE | |
| 2492 | 16216 | 031477 | | STA TEMP4 | 'N' WORD TO 0100 | |
| 2493* | | | | | | |
| 2494 | 16217 | 021477 | CONT | LDA TEMP4 | IF THE LOW DIGIT OF THE | |
| 2495 | 16220 | 070111 | | SLA *+2 | 'N' WORD IS A 1 THEN | |
| 2496 | 16221 | 045476 | | ISZ TEMP3 | INCREMENT POINTER 2 | |
| 2497 | 16222 | 070146 | | RAR 4 | ROTATE THE 'N' WORD ONE | |
| 2498 | 16223 | 031477 | | STA TEMP4 | BCD AND REPLACE | |
| 2499* | | | | | | |
| 2500 | 16224 | 021475 | | LDA TEMP2 | MASK OUT THE LAST | |
| 2501 | 16225 | 070056 | | CMA | FIVE FROM THE | |
| 2502 | 16226 | 151701 | | AND TEMP4,I | WORKING | |
| 2503 | 16227 | 131701 | | STA TEMP4,I | REGISTER | |
| 2504* | | | | | | |
| 2505 | 16230 | 021475 | | LDA TEMP2 | IF THE LOW DIGIT OF THE | |

340

328

1,444,141

328

| PAGE 224 | | | MORE 9830 FUNCTIONS | |
|----------|-------|--------|---------------------|-----------------------------------|
| 2506 | 16231 | 070111 | SLA *+2 | 'FIVE' WORD IS A 5 THEN |
| 2507 | 16232 | 045701 | ISZ TEMP A | INCREMENT POINTER 1 |
| 2508 | 16233 | 070146 | RAR 4 | ROTATE THE 'FIVE' WORD |
| 2509 | 16234 | 031475 | STA TEMP2 | ONE BCD AND REPLACE |
| 2510 | 16235 | 101701 | ADA TEMP A,I | LOCATE THE NEXT FIVE INTO |
| 2511 | 16236 | 131701 | STA TEMP A,I | THE WORKING REGISTER |
| 2512* | | | | |
| 2513 | 16237 | 125476 | LDB TEMP3,I | LOAD THE WORKING REGISTER WORD |
| | | | | TO BE INCREMENTED INT |
| 2514 | 16240 | 170400 | CMY | COMPLEMENT THE ARGUMENT |
| 2515* | | | | |
| 2516 | 16241 | 020061 | LDA P10 | SET THE INCREMENT |
| 2517 | 16242 | 031466 | STA TEMP5 | COUNTER=10 |
| 2518* | | | | |
| 2519 | 16243 | 135476 | STB TEMP3,I | STORE B INTO THE WORKING REGISTER |
| 2520 | 16244 | 170560 | FXA | ADD THE WORKING REGISTER TO THE |
| | | | | ARGUMENT |
| 2521 | 16245 | 005477 | ADB TEMP4 | INCREMENT THE PROPER DIGIT OF B |
| 2522 | 16246 | 055466 | DSZ TEMP5 | DECREMENT THE INCREMENT COUNTER |
| | | | | AND EXIT LOOP IF ZER |
| 2523 | 16247 | 071635 | SEC *-4,C | IF NO OVERFLOW OCCURRED JUMP |
| | | | | BACK INTO LOOP |
| 2524* | | | | |
| 2525 | 16250 | 170400 | CMY | COMPLEMENT THE ARGUMENT |
| 2526* | | | | |
| 2527 | 16251 | 021755 | LDA AR2E+1 | JUMP OUT OF THE |
| 2528 | 16252 | 001756 | ADA AR2E+2 | MAIN LOOP IF |
| 2529 | 16253 | 001757 | ADA AR2E+3 | AN PERFECT SQUARE |
| 2530 | 16254 | 070134 | SES *-2,C | HAS BEEN |
| 2531 | 16255 | 070350 | SZA PFSO | FOUND |
| 2532* | | | | |
| 2533 | 16256 | 170560 | FXA | RESTORE THE ARGUMENT |
| 2534 | 16257 | 171400 | MLS | SHIFT THE ARGUMENT LEFT ONCE |
| 2535* | | | | |
| 2536 | 16260 | 021476 | LDA TEMP3 | IF 12 DIGITS HAVE |
| 2537 | 16261 | 010057 | CPA ATEMP | BEN CALCULATED |
| 2538 | 16262 | 066302 | JMP LSTLP | EXIT FROM MAIN LOOP |
| 2539 | 16263 | 066217 | JMP CONT | ELSE CONTINUE |
| 2540* | | | | |
| 2541 | 16264 | 025701 | PFSQ LDB TEMP A | A PERFECT SQUARE HAS BEEN FOUND |
| 2542 | 16265 | 014057 | CPB .1000 | |
| 2543 | 16266 | 066272 | JMP *-4 | DO NOT STORE THE LAST DIGIT |
| 2544 | 16267 | 004025 | ADB P8 | SO INCREMENT THE LEAST |
| | | | | SIGNIFICANT |
| 2545 | 16270 | 021475 | LDA TEMP2 | DIGIT (A5) OF THE WORKING |
| | | | | REGISTER |
| 2546 | 16271 | 074557 | STA B,I | BY 5 AND |
| 2547 | 16272 | 170560 | FXA | THEN LEFT SHIFT |
| 2548 | 16273 | 171400 | MLS | IT ONCE |
| 2549* | | | | |
| 2550 | 16274 | 070250 | SZA *-5 | IF A DIGIT WAS SHIFTED OU |
| 2551 | 16275 | 070137 | LDB A | |
| 2552 | 16276 | 174470 | MRY | SHIFT IT BACK IN AND INCREMENT |
| | | | | THE EXPONENT |
| 2553 | 16277 | 024013 | LDB P256 | |
| 2554 | 16300 | 065237 | JMP STAR2 | |
| 2555 | 16301 | 065240 | JMP STAR2+1 | OTHERWISE STORE THE ANSWER AND |
| | | | | RETURN |
| 2556* | | | | |
| 2557 | 16302 | 021750 | LSTLP LDA AR1E+4 | LOCATE THE LAST DIGIT |
| 2558 | 16303 | 070542 | SAR 12 | ALIGN IT |
| 2559 | 16304 | 175400 | DLS | AND SHIFT IT INTO THE ANSWER |
| 2560 | 16305 | 020213 | LDA AR1A | |
| 2561 | 16306 | 024214 | LDB AR2A | |

341

```
2562 16307 170004 XFR
2563 16310 065254 JMP ZONK
2564*
2565*
2566*****
2567*
2568* FLOATING POINT MULTIPLY
2569*
2570*
2571 16311 031702 FMP STA ADR1
2572 16312 074117 LDA B
2573 16313 024213 LDB AR1A TRANSFER THE MULTIPLICAND INTO A
2574 16314 170004 XFR TEMPORARY
2575*
2576 16315 021702 LDA ADR1 TRANSFER THE MULTIPLIER
2577 16316 024057 LDB ATEMP INTO A TEMPORARY
2578 16317 170004 XFR
2579 16320 020214 LDA AR2A
2580 16321 170000 CLR INITIALIZE WORD POINTER FOR THE
2581 16322 031475 STA TEMP2 MULTIPLIER AND CLEAR AR2
2582*
2583 16323 020134 LDA M3 INITIALIZE THE
2584 16324 031476 STA TEMP3 WORD COUNT TO 3
2585*
2586 16325 055475 NXWD DSZ TEMP2 DECREMENT THE WORD POINTER
2587 16326 125475 LDB TEMP2,I LOAD 4 DIGITS OF THE MULTIPLIER
2588*
2589 16327 171460 FMP ACCUMULATE THE PRESENT DIGIT
MULTIPLICAND
2590 16330 004145 ADB N14 SET BITS 0-3=0001 IN B
2591 16331 174470 MRY ALIGN THE TEMP ANSWER FOR NEXT
DIGIT
2592 16332 074142 SBR 4 SHIFT IN THE NEXT MULTIPLIER DIGIT
2593*
2594 16333 171460 FMP
2595 16334 004145 ADB N14
2596 16335 174470 MRY
2597 16336 074142 SBR 4
2598*
2599 16337 171460 FMP
2600 16340 004145 ADB N14
2601 16341 174470 MRY
2602 16342 074142 SBR 4
2603*
2604 16343 171460 FMP
2605 16344 004145 ADB N14
2606 16345 174470 MRY
2607*
2608 16346 045476 ISZ TEMP3 INCREMENT THE WORD COUNT AND
2609 16347 066325 JMP NXWD JUMP IF NON ZERO
2610*
2611 16350 025755 LDB AR2E+1 SKIP IF THE FIRST
2612 16351 074542 SBR 12 DIGIT OF THE ANSWER
2613 16352 076350 RZB *+7 IS NOT ZERO
2614*
2615 16353 070137 LDB A IT IS ZERO SO
2616 16354 171400 MLS NORMALIZE ANSWER
2617 16355 005757 ADB AR2E+3 PICK UP THE 12TH DIGIT
```

PAGE 226

MORE 9830 FUNCTIONS

2618 16356 035757 STB AR2E+3 WHICH WAS SHIFTED OUT ABOVE
 2619*
 2620 16357 124057 LDB ATEMP,I LOAD THE EXPONENT WORD OF THE
 2621 16360 065237 JMP STAR2 MULTIPLIER AND EXIT TO RETURN
 ANSWER
 2622*
 2623 16361 061307 JSM ROUND+2 ROUND UP ANSWER IF 13TH DIGIT
 IS >=5
 2624 16362 124057 LDB ATEMP,I LOAD EXPONENT OF THE MULTIPLIER
 2625 16363 004013 ADB P256 INCREMENT IT
 2626 16364 065237 JMP STAR2 EXIT TO RETURN ANSWER
 2627*
 2628*
 2629* CONVERT SPLIT TO FULL-PRECISION FLOATING POINT NUMBER
 2630*
 2631* LDB ADDRESS OF SPLIT
 2632* JSM SPLFL
 2633*
 2634* FULL FLOATING POINT NUMBER IN AR2 ON RETURN
 2635*
 2636 16365 061256 SPLFL JSM CLAR2 CLEAR AR2
 2637 16366 074517 LDA B,I LOAD EXPONENT, SIGN, D1 AND D2
 2638 16367 031754 STA AR2 SAVE TEMPORARLY IN AR2
 2639 16370 050105 AND B377 MASK D1 AND D2
 2640 16371 031755 STA AR2+1 AND STORE
 2641 16372 074070 SIB *+1
 2642 16373 074517 LDA B,I
 2643 16374 031756 STA AR2+2 STORE D3 THUR D6
 2644 16375 171450 NRM
 2645 16376 074742 SBR 16
 2646 16377 021754 LDA AR2 FIX
 2647 16400 070340 AAR 8 UP
 2648 16401 070131 SLA *+2,C THE
 2649 16402 074070 SIB *+1 FULL
 2650 16403 070444 SAL 7 EXPONENT
 2651 16404 074217 IOR B
 2652 16405 031754 STA AR2
 2653 16406 170402 RET
 2654*
 2655 16407 000000 BSS 1 ***CHECKSUM WORD***
 2656*
 2657 END
 * NO ERRORS *

PAGE 227

CROSS-REFERENCE TABLE

| | | | | |
|-------|------|------|------|------|
| .1 | 0020 | 0522 | 2063 | 2311 |
| .10 | 0071 | 0072 | 0574 | 1934 |
| .100 | 0070 | 0685 | | |
| .1000 | 0069 | 0534 | 2542 | |
| .11 | 0035 | 0043 | | |
| .12 | 0036 | 0527 | 2195 | |
| .13 | 0039 | | | |
| .15 | 0048 | 0083 | 0529 | |
| .16 | 0049 | 0084 | | |
| .1E4 | 0068 | | | |
| .2 | 0027 | 0523 | 2077 | 2266 |
| .21 | 0050 | | | |
| .22 | 0051 | 0087 | 0684 | |
| .23 | 0052 | 0088 | | |
| .28 | 0053 | | | |
| .3 | 0028 | 0045 | | |
| .31 | 0054 | 0090 | | |
| .32 | 0055 | | | |
| .33 | 0056 | | | |
| .34 | 0057 | 0091 | | |
| .37 | 0058 | | | |
| .4 | 0029 | 0524 | 2161 | |
| .40 | 0059 | | | |
| .41 | 0060 | | | |
| .43 | 0061 | | | |
| .45 | 0062 | | | |
| .46 | 0063 | 1915 | 2137 | |
| .48 | 0064 | | | |
| .5 | 0030 | 0525 | 2205 | 2253 |
| .58 | 0065 | 0094 | | |
| .6 | 0031 | 0044 | 0572 | |
| .63 | 0066 | | | |
| .7 | 0032 | 0573 | | |
| .8 | 0033 | 0526 | | |
| .9 | 0034 | | | |
| .99 | 0074 | | | |
| .ANUM | 2108 | 2121 | | |
| .BUFA | 0180 | | | |
| .CS | 0989 | 0982 | | |
| .EXP | 1262 | 1908 | 2332 | |
| .FAD | 0930 | 1388 | | |
| .FAD. | 1780 | 1767 | | |
| .FADA | 0185 | 0582 | | |
| .FDV | 0920 | 0836 | 0987 | 1056 |
| .FDVA | 0188 | 0584 | 2416 | 1162 |
| .FMP | 0952 | 0794 | 0856 | 1260 |
| .FMFA | 0187 | 0583 | 2391 | 2404 |
| .FSBA | 0186 | 2302 | | |
| .FSR | 0190 | 0866 | | |
| .GOON | 1879 | 1871 | | |
| .IK | 2030 | 2020 | 2046 | |
| .LNUM | 0314 | | | |
| .NPWR | 0504 | | | |
| .RI | 1982 | 2013 | | |
| .SIGN | 0399 | 0400 | 2155 | |

| | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|
| .SLBF | 0181 | | | | | | | | | | | |
| .T. | 1138 | 1128 | | | | | | | | | | |
| .TEMP | 0350 | 0351 | | | | | | | | | | |
| .UPAI | 2421 | 2364 | | | | | | | | | | |
| A | 0000 | 1044 | 1147 | 1284 | 1286 | 1366 | 1715 | 1742 | 1776 | 1940 | 1942 | |
| | 2096 | 2255 | 2551 | 2615 | | | | | | | | |
| ABS | 2043 | 1906 | | | | | | | | | | |
| ACS | 1028 | 1892 | | | | | | | | | | |
| ACSC | 1035 | 1031 | | | | | | | | | | |
| ACTSA | 0174 | | | | | | | | | | | |
| ADS | 0710 | 2489 | | | | | | | | | | |
| ADD | 0582 | 0932 | | | | | | | | | | |
| ADR0 | 0694 | 1341 | 1378 | | | | | | | | | |
| ADR1 | 0549 | 0746 | 0750 | 0793 | 0930 | 1694 | 1709 | 1710 | 1727 | 1792 | 1798 | |
| | 2354 | 2357 | 2362 | 2368 | 2571 | 2576 | | | | | | |
| ADR2 | 0550 | 0551 | 0726 | 0727 | 0730 | 0732 | 0920 | 0931 | 1259 | 1691 | 1708 | |
| | 1713 | 1718 | 1793 | 1879 | 2382 | 2421 | 2429 | 2442 | 2446 | | | |
| ADR3 | 0551 | 0835 | 0919 | 0934 | 1057 | 1177 | 1261 | 1376 | | | | |
| ADRSS | 0327 | | | | | | | | | | | |
| ADRX | 0692 | 0848 | 0851 | 0872 | 0876 | 1445 | | | | | | |
| ADRY | 0693 | 0694 | 1443 | | | | | | | | | |
| ADRYE | 0565 | | | | | | | | | | | |
| ADRYE | 2329 | 2380 | 2390 | 2392 | 2412 | 2415 | | | | | | |
| AERRA | 0176 | 1324 | 1810 | 2056 | 2071 | 2342 | | | | | | |
| ARI | 0413 | 0182 | 0421 | 2180 | 2181 | 2182 | 2296 | 2313 | 2366 | | | |
| AR1A | 0182 | 0961 | 1154 | 1158 | 1416 | 1531 | 1566 | 1593 | 1638 | 1649 | 1722 | |
| | 1724 | 1745 | 1799 | 2309 | 2456 | 2475 | 2560 | 2573 | | | | |
| AR1E | 0421 | 1174 | 1734 | 1753 | 1763 | 1772 | 1808 | 1880 | 1883 | 2461 | 2467 | |
| | 2481 | 2487 | 2557 | | | | | | | | | |
| AR2 | 0415 | 0183 | 0422 | 0999 | 1930 | 1988 | 1990 | 2016 | 2027 | 2028 | 2044 | |
| | 2062 | 2065 | 2075 | 2080 | 2084 | 2087 | 2114 | 2139 | 2160 | 2208 | 2210 | |
| AR2A | 2268 | 2270 | 2271 | 2272 | 2278 | 2436 | 2638 | 2640 | 2643 | 2646 | 2652 | |
| | 0183 | 0736 | 0753 | 0768 | 0847 | 0855 | 0865 | 0933 | 0949 | 1342 | 1465 | |
| | 1473 | 1488 | 1501 | 1505 | 1530 | 1535 | 1565 | 1592 | 1613 | 1720 | 1726 | |
| | 1746 | 1795 | 1867 | 2216 | 2246 | 2258 | 2300 | 2393 | 2406 | 2458 | 2561 | |
| AR2E | 0422 | 0766 | 0767 | 0778 | 0789 | 0801 | 0819 | 0820 | 0821 | 0825 | 0989 | |
| | 1007 | 1077 | 1101 | 1111 | 1121 | 1125 | 1142 | 1146 | 1173 | 1200 | 1207 | |
| | 1221 | 1225 | 1306 | 1367 | 1370 | 1459 | 1731 | 1754 | 1773 | 1780 | 1812 | |
| | 1816 | 1836 | 1837 | 1838 | 1873 | 1875 | 1876 | 2527 | 2528 | 2529 | 2611 | |
| | 2617 | 2618 | | | | | | | | | | |
| ARYAD | 0403 | | | | | | | | | | | |
| ASN | 1065 | 1893 | | | | | | | | | | |
| ASNS | 1042 | 1029 | 1065 | | | | | | | | | |
| ATD | 1483 | 1119 | 1290 | | | | | | | | | |
| ATEMP | 0534 | 1693 | 1696 | 1699 | 1802 | 1865 | 2383 | 2389 | 2402 | 2403 | 2405 | |
| | 2537 | 2577 | 2620 | 2624 | | | | | | | | |
| ATN | 1242 | 1899 | | | | | | | | | | |
| ATNS | 1187 | 1058 | 1242 | | | | | | | | | |
| AWRDN | 0691 | 0884 | 1483 | 1552 | | | | | | | | |
| B | 0000 | 0805 | 1049 | 1519 | 1692 | 2103 | 2190 | 2200 | 2212 | 2218 | 2262 | |
| | 2546 | 2572 | 2637 | 2642 | 2651 | | | | | | | |
| B100 | 0095 | | | | | | | | | | | |
| B1000 | 0102 | 2194 | | | | | | | | | | |
| B10K | 0114 | 0115 | 0535 | | | | | | | | | |
| B12K | 0116 | 0117 | 0206 | | | | | | | | | |

CROSS-REFERENCE TABLE

[illegible]

CROSS-REFERENCE TABLE

[illegible]

PAGE 232

CROSS-REFERENCE TABLE

| | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|
| FXFL | 2019 | 2039 | | | | | | | |
| FXFLA | 0194 | | | | | | | | |
| FXFLT | 2012 | 0194 | 1976 | | | | | | |
| GET | 1115 | 1081 | 1108 | 1114 | | | | | |
| GET3 | 1080 | 1086 | | | | | | | |
| GETAD | 0471 | | | | | | | | |
| GETCH | 0481 | | | | | | | | |
| GETSK | 0485 | | | | | | | | |
| GFG | 0554 | 0901 | 1030 | 1045 | 1080 | 1172 | | | |
| GFLAG | 0404 | 2167 | | | | | | | |
| GFRST | 0484 | | | | | | | | |
| GNEXT | 0483 | 2115 | | | | | | | |
| GONO | 1235 | 1222 | | | | | | | |
| GOOD | 1816 | 1809 | | | | | | | |
| GOON | 1592 | 1571 | 1589 | | | | | | |
| GPARM | 0434 | | | | | | | | |
| GPERR | 0683 | 0910 | 1247 | | | | | | |
| GRA | 0706 | 1248 | | | | | | | |
| GRAF | 0700 | 0852 | | | | | | | |
| GSIGN | 0496 | 2168 | | | | | | | |
| GTCON | 0510 | | | | | | | | |
| GTMP | 0482 | | | | | | | | |
| GTOPP | 0476 | | | | | | | | |
| GTOR | 0910 | | | | | | | | |
| GTSTA | 0213 | | | | | | | | |
| HFLGI | 0321 | | | | | | | | |
| HSTPT | 0302 | | | | | | | | |
| I | 0067 | | | | | | | | |
| IEXP | 2332 | 2432 | | | | | | | |
| ICNT | 0432 | | | | | | | | |
| IINIT | 2331 | 2356 | | | | | | | |
| ILNX | 2330 | 2430 | | | | | | | |
| IMPY | 0497 | 1935 | | | | | | | |
| IN | 1401 | 1140 | 1294 | | | | | | |
| IN2 | 1439 | 1150 | 1206 | | | | | | |
| IN3 | 1609 | 1218 | 1339 | | | | | | |
| INF | 0077 | | | | | | | | |
| INIT | 0724 | 0903 | 1046 | 1188 | 1263 | 1321 | 2331 | | |
| INITS | 0466 | | | | | | | | |
| INREA | 0226 | | | | | | | | |
| INT | 2288 | 1904 | 2295 | 2299 | | | | | |
| INTCK | 0469 | 2174 | | | | | | | |
| INTEA | 0178 | | | | | | | | |
| INTEL | 0340 | | | | | | | | |
| INTFL | 0075 | | | | | | | | |
| INTGR | 0402 | | | | | | | | |
| INTLK | 1769 | 1755 | | | | | | | |
| IOBUF | 0359 | | | | | | | | |
| IOHEA | 0229 | | | | | | | | |
| IOSTA | 0225 | | | | | | | | |
| IPW1 | 2386 | 2408 | | | | | | | |
| IPW2 | 2397 | 2387 | | | | | | | |
| IPWR | 2375 | | | | | | | | |
| ITR | 0752 | 0833 | 0868 | 1032 | 1131 | | | | |
| KEYFG | 0324 | | | | | | | | |
| L2TI | 0347 | 1977 | 1979 | 1986 | 1989 | 2021 | 2130 | 2250 | 2274 |

CROSS-REFERENCE TABLE

| | | | | | |
|-------|------|------|------|------|------|
| L2T2 | 0348 | 1916 | 1933 | 1944 | 1960 |
| LASTN | 0398 | | | | |
| LBOP | 0119 | | | | |
| LDEF | 0499 | | | | |
| LDVSR | 1915 | 2017 | | | |
| LETCK | 0474 | | | | |
| LIMIT | 0475 | | | | |
| LISTA | 0167 | | | | |
| LN10 | 0688 | 1267 | 1337 | 1373 | 1670 |
| LNFD | 0072 | | | | |
| LNINC | 0315 | | | | |
| LNUM | 0467 | | | | |
| LNx | 1320 | 1669 | 1907 | 2330 | |
| LOCL1 | 0345 | 0542 | | | |
| LOCL2 | 0346 | 2031 | 2034 | | |
| LOG | 1669 | 1898 | | | |
| LPCK | 0447 | | | | |
| LPCKE | 0448 | | | | |
| LPOP | 0120 | | | | |
| LROMA | 0123 | | | | |
| LSTAK | 0299 | | | | |
| LSTLP | 2557 | 2538 | | | |
| LSTPT | 0301 | | | | |
| LTR | 0489 | | | | |
| LWAM | 0295 | | | | |
| M1 | 0124 | 0531 | 2032 | 2098 | 2166 |
| M10 | 0131 | 0532 | | | |
| M100 | 0144 | 0687 | 2173 | | |
| M1024 | 0146 | | | | |
| M11 | 0132 | 1009 | 2024 | 2316 | |
| M12 | 0133 | 0577 | 2101 | 2120 | |
| M13 | 0134 | 0686 | 1822 | | |
| M14 | 0135 | 0711 | | | |
| M15 | 0136 | | | | |
| M2 | 0125 | 0575 | 2163 | | |
| M23 | 0137 | | | | |
| M25 | 0138 | | | | |
| M256 | 0145 | 0533 | 1932 | 1980 | 2209 |
| M3 | 0126 | 2583 | | | 2273 |
| M32 | 0139 | | | | |
| M36 | 0140 | | | | |
| M4 | 0127 | 0576 | 1825 | | |
| M48 | 0141 | | | | |
| M5 | 0128 | 0423 | 1964 | | |
| M6 | 0129 | | | | |
| M64 | 0142 | | | | |
| M80 | 0143 | | | | |
| M9 | 0130 | | | | |
| MASK1 | 0153 | | | | |
| MASK2 | 1684 | 1743 | | | |
| MASK5 | 0154 | | | | |
| MAW | 0294 | | | | |
| MAXIN | 0400 | | | | |
| MAXSN | 0147 | | | | |
| MBIN1 | 0406 | 2064 | 2082 | | |
| MBIN2 | 0407 | 2061 | 2085 | 2086 | 2088 |

PAGE 235

CROSS-REFERENCE TABLE

[illegible]



PAGE 236

CROSS-REFERENCE TABLE

| | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|
| PTH | 1575 | 1568 | 1597 | | | | | | | | | |
| PTI | 1570 | 1562 | | | | | | | | | | |
| PTJ | 1552 | 1303 | 1333 | | | | | | | | | |
| PTK | 1534 | 1521 | | | | | | | | | | |
| PTL | 1485 | 1450 | | | | | | | | | | |
| PTL1 | 1450 | 1457 | | | | | | | | | | |
| PTM | 1497 | 1481 | | | | | | | | | | |
| PTN | 1469 | 1452 | 1463 | | | | | | | | | |
| PTO | 1459 | | | | | | | | | | | |
| PTP | 1616 | 1626 | | | | | | | | | | |
| PTQ | 1635 | 1662 | | | | | | | | | | |
| PTR | 1561 | 1600 | | | | | | | | | | |
| PTS | 1565 | 1558 | | | | | | | | | | |
| QTCHA | 0230 | | | | | | | | | | | |
| QTOP | 0109 | 0161 | | | | | | | | | | |
| RAD | 0705 | 0907 | | | | | | | | | | |
| RAX | 0875 | 0871 | | | | | | | | | | |
| RBOP | 0117 | | | | | | | | | | | |
| RBUFF | 0360 | | | | | | | | | | | |
| RCOUT | 0333 | | | | | | | | | | | |
| RCPFG | 0557 | 0991 | 1076 | 1093 | 1097 | 1132 | 1136 | 1152 | 1165 | 1167 | 1287 | |
| | 1305 | | | | | | | | | | | |
| RDYDA | 0166 | | | | | | | | | | | |
| RECIP | 0833 | 1135 | 1197 | 1311 | | | | | | | | |
| REED | 0161 | | | | | | | | | | | |
| REEDL | 0339 | | | | | | | | | | | |
| RES | 0417 | | | | | | | | | | | |
| RESB1 | 0463 | | | | | | | | | | | |
| RESBP | 0464 | | | | | | | | | | | |
| RET1 | 1958 | 1943 | | | | | | | | | | |
| RET2 | 0431 | | | | | | | | | | | |
| RET4 | 2336 | 2360 | | | | | | | | | | |
| RETN2 | 0430 | 1948 | 2164 | 2326 | | | | | | | | |
| RETN3 | 0429 | | | | | | | | | | | |
| ROUND | 0425 | 0810 | 1117 | 1486 | 1523 | 1751 | 2473 | 2623 | | | | |
| RPCK | 0449 | | | | | | | | | | | |
| RPCKE | 0450 | | | | | | | | | | | |
| RPOP | 0115 | | | | | | | | | | | |
| RSAGE | 0292 | | | | | | | | | | | |
| RSLT | 0696 | 0731 | 0735 | 0752 | 0860 | 0873 | 0875 | 0929 | 1160 | 1176 | 1381 | |
| | 1387 | | | | | | | | | | | |
| RSLTE | 0373 | 0696 | | | | | | | | | | |
| RSRTR | 0418 | | | | | | | | | | | |
| RSTAK | 0416 | | | | | | | | | | | |
| RSTKA | 0158 | | | | | | | | | | | |
| RSULT | 0436 | | | | | | | | | | | |
| RTOD | 0944 | 1246 | 1249 | | | | | | | | | |
| RTT | 0735 | 0762 | 0966 | | | | | | | | | |
| RUNA | 0163 | | | | | | | | | | | |
| RY1 | 0559 | 1155 | 1157 | 1444 | 1466 | 1472 | 1489 | 1497 | 1500 | 1534 | | |
| RY2 | 0561 | 1446 | 1498 | 1499 | 1504 | | | | | | | |
| S | 0082 | | | | | | | | | | | |
| SALTA | 0207 | | | | | | | | | | | |
| SANS | 0933 | 0922 | 0953 | 0983 | | | | | | | | |
| SAVBP | 0462 | | | | | | | | | | | |
| SAVE | 0560 | 1208 | 1226 | 1439 | 1516 | 1517 | 2312 | 2322 | 2324 | 2367 | 2375 | |

| | | | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|--|--|
| | 2410 | 2428 | 2434 | 2445 | | | | | | | | | | |
| SAVEA | 0325 | | | | | | | | | | | | | |
| SAVEB | 0326 | | | | | | | | | | | | | |
| SBADR | 0394 | | | | | | | | | | | | | |
| SBPTR | 0387 | 2178 | 2201 | 2202 | 2203 | | | | | | | | | |
| SBPUD | 0441 | 2213 | 2219 | | | | | | | | | | | |
| SBUFA | 0199 | | | | | | | | | | | | | |
| SCRT | 0845 | 0985 | 1047 | | | | | | | | | | | |
| SCRT1 | 0870 | 1055 | | | | | | | | | | | | |
| SEC | 0320 | | | | | | | | | | | | | |
| SFLAG | 0393 | | | | | | | | | | | | | |
| SGN | 2004 | 1903 | | | | | | | | | | | | |
| SGNS | 0503 | 2004 | | | | | | | | | | | | |
| SHIFT | 0555 | 1402 | 1429 | 1440 | 1509 | 1542 | 1556 | 1575 | 1599 | 1611 | 1625 | | | |
| | 1632 | 1641 | 1661 | | | | | | | | | | | |
| SIGDG | 2180 | 2102 | 2119 | 2127 | 2129 | 2196 | | | | | | | | |
| SIGN | 0562 | 0728 | 0994 | 1052 | 1078 | 1110 | 1112 | 1115 | 1164 | 1220 | 1274 | | | |
| | 1309 | 1347 | 1383 | | | | | | | | | | | |
| SIN | 0976 | 1014 | 1902 | | | | | | | | | | | |
| SLCOD | 0328 | | | | | | | | | | | | | |
| SLEND | 0202 | | | | | | | | | | | | | |
| SLWST | 0457 | | | | | | | | | | | | | |
| SNFLA | 0197 | | | | | | | | | | | | | |
| SNFLT | 2244 | 0197 | | | | | | | | | | | | |
| SOUT | 1251 | 1312 | | | | | | | | | | | | |
| SPLFA | 0196 | | | | | | | | | | | | | |
| SPLFL | 2636 | 0196 | | | | | | | | | | | | |
| SPLUF | 2056 | 2068 | | | | | | | | | | | | |
| SSBPT | 0395 | | | | | | | | | | | | | |
| SSYMA | 0170 | | | | | | | | | | | | | |
| ST | 0377 | | | | | | | | | | | | | |
| STAR2 | 0424 | 1179 | 1782 | 1884 | 2554 | 2555 | 2621 | 2626 | | | | | | |
| START | 0162 | 0007 | | | | | | | | | | | | |
| STEXP | 0453 | | | | | | | | | | | | | |
| STM1 | 1276 | 1265 | 1273 | | | | | | | | | | | |
| STMAX | 0427 | 1103 | 1328 | 1814 | 2073 | 2341 | | | | | | | | |
| STN | 0900 | 0976 | 1006 | 1074 | | | | | | | | | | |
| STOR1 | 0438 | | | | | | | | | | | | | |
| STORE | 0437 | | | | | | | | | | | | | |
| STPFL | 0323 | | | | | | | | | | | | | |
| STROP | 0440 | | | | | | | | | | | | | |
| STSRH | 0452 | | | | | | | | | | | | | |
| SUMPA | 0227 | | | | | | | | | | | | | |
| SYE12 | 0495 | | | | | | | | | | | | | |
| SYE25 | 0470 | | | | | | | | | | | | | |
| SYMCI | 0443 | | | | | | | | | | | | | |
| SYMC2 | 0442 | | | | | | | | | | | | | |
| SYMCK | 0444 | | | | | | | | | | | | | |
| SYMTP | 0298 | | | | | | | | | | | | | |
| SYTSA | 0209 | | | | | | | | | | | | | |
| TABL0 | 0553 | 1401 | 1414 | 1415 | 1574 | 1588 | 1609 | 1636 | 1637 | | | | | |
| TAN | 1074 | 0978 | 1900 | | | | | | | | | | | |
| TEMP | 0552 | 0553 | 0554 | 0555 | 0556 | 0557 | 0558 | 0559 | 0560 | 0561 | 0562 | | | |
| | 0563 | 0564 | 0902 | 0977 | 1102 | 1610 | 1629 | | | | | | | |
| TEMP1 | 0352 | 2189 | 2222 | | | | | | | | | | | |
| TEMP2 | 0353 | 1803 | 1856 | 1857 | 1864 | 2490 | 2500 | 2505 | 2509 | 2545 | 2581 | | | |

PAGE 239

9830

ASMB,A,B,L

```

0001
0003*
0004*
0005*
0006*
0007*
0008*
0009*
0010*
0011 00000
0012*
0013 00000 164121
0014 00001 043060
0015 00002 000000
0016*
0017 00003 000000 C6
0018 00004 010000
0019 00005 000000
0020*
0021 00006 000000 C5
0022 00007 012560
0023 00010 074543
0024 00011 023200
0025*
0026 00012 000001 .1
0027*
0028 00013 000400 C7
0029 00014 053451
0030 00015 053571
0031 00016 050450
0032*
0033 00017 000002 .2
0034 00020 000003 .3
0035 00021 000004 .4
0036 00022 000005 .5
0037 00023 000006 .6
0038 00024 000007 .7
0039 00025 000010 .8
0040 00026 000011 .9
0041 00027 000013 .11
0042 00030 000014 .12
0043 00031 000015 CRTN
0044 00031 EOL
0045 00031 .13
0046*****
0047* POINTERS TO FUNCTIONS CONSTANTS
0048*
0049 00027 DPERR EQU .11
0050 00023 PI/2 EQU .6
0051 00020 ONE EQU .3
0052*
0053*****
0054 00032 000017 .15
0055 00033 000020 .16
0056 00034 000025 .21
0057 00035 000026 .22

```

ORG 0
 JMP START,I
 OCT 43060
 BSS 1
 START OF MONITOR
 CONSTANT NEEDED BY PROCESSOR
 FOR INTERPT
 ONE
 0000
 0000
 PI/2
 1570
 7963
 2680
 DEC 1
 OCT 000400 180/PI
 OCT 053451 5729
 OCT 053571 5779
 OCT 050450 5128
 DEC 2
 DEC 3
 DEC 4
 DEC 5
 DEC 6
 DEC 7
 DEC 8
 DEC 9
 DEC 11
 DEC 12
 OCT 15
 EQU CRTN
 EQU CRTN
 POINTER TO 180/PI
 POINTER TO PI/2
 POINTER TO THE CONSTANT ONE
 DEC 15
 DEC 16
 DEC 21
 DEC 22

PAGE 240

9830

| | | | | | |
|------|-------|--------|-------|-----|--------|
| 0058 | 00036 | 000027 | .23 | DEC | 23 |
| 0059 | 00037 | 000034 | .28 | DEC | 28 |
| 0060 | 00040 | 000037 | .31 | DEC | 31 |
| 0061 | 00041 | 000040 | .32 | DEC | 32 |
| 0062 | 00042 | 000041 | .33 | DEC | 33 |
| 0063 | 00043 | 000042 | .34 | DEC | 34 |
| 0064 | 00044 | 000045 | .37 | DEC | 37 |
| 0065 | 00045 | 000050 | .40 | DEC | 40 |
| 0066 | 00046 | 000051 | .41 | DEC | 41 |
| 0067 | 00047 | 000053 | .43 | DEC | 43 |
| 0068 | 00050 | 000055 | .45 | DEC | 45 |
| 0069 | 00051 | 000056 | .46 | DEC | 46 |
| 0070 | 00052 | 000060 | .48 | DEC | 48 |
| 0071 | 00053 | 000072 | .58 | DEC | 58 |
| 0072 | 00054 | 000077 | .63 | DEC | 63 |
| 0073 | 00055 | 000111 | I | OCT | 111 |
| 0074 | 00056 | 023420 | .1E4 | DEC | 10000 |
| 0075 | 00057 | 001750 | .1000 | DEC | 1000 |
| 0076 | 00060 | 000144 | .100 | DEC | 100 |
| 0077 | 00061 | 000012 | .10 | DEC | 10 |
| 0078 | 00061 | | LNFD | EQU | .10 |
| 0079 | 00062 | 100000 | FLGBT | OCT | 100000 |
| 0080 | 00063 | 000143 | .99 | DEC | 99 |
| 0081 | 00064 | 100004 | INTFL | OCT | 100004 |
| 0082 | 00065 | 076000 | OPMSK | OCT | 76000 |
| 0083 | 00066 | 077777 | INF | OCT | 77777 |
| 0084 | 00067 | 100037 | TYPFL | OCT | 100037 |
| 0085 | 00070 | 101777 | OPDMK | OCT | 101777 |
| 0086 | 00071 | 000105 | E | OCT | 105 |
| 0087 | 00072 | 000106 | F | OCT | 106 |
| 0088 | 00073 | 000123 | S | OCT | 123 |
| 0089 | 00032 | | B17 | EQU | .15 |
| 0090 | 00033 | | B20 | EQU | .16 |
| 0091 | 00074 | 000022 | B22 | OCT | 22 |
| 0092 | 00075 | 000023 | B23 | OCT | 23 |
| 0093 | 00035 | | B26 | EQU | .22 |
| 0094 | 00036 | | B27 | EQU | .23 |
| 0095 | 00076 | 000036 | B36 | OCT | 36 |
| 0096 | 00040 | | B37 | EQU | .31 |
| 0097 | 00043 | | B42 | EQU | .34 |
| 0098 | 00077 | 000044 | B44 | OCT | 44 |
| 0099 | 00100 | 000052 | B52 | OCT | 52 |
| 0100 | 00053 | | B72 | EQU | .58 |
| 0101 | 00101 | 000100 | B100 | OCT | 100 |
| 0102 | 00102 | 000133 | B133 | OCT | 133 |
| 0103 | 00103 | 000140 | B140 | OCT | 140 |
| 0104 | 00104 | 000177 | B177 | OCT | 177 |
| 0105 | 00105 | 000377 | B377 | OCT | 377 |
| 0106 | 00013 | | B400 | EQU | C7 |
| 0107 | 00106 | 000777 | B777 | OCT | 777 |
| 0108 | 00107 | 001000 | B1000 | OCT | 1000 |
| 0109 | 00110 | 001400 | B1400 | OCT | 1400 |
| 0110 | 00111 | 001740 | B1740 | OCT | 1740 |
| 0111 | 00112 | 001760 | B1760 | OCT | 1760 |
| 0112 | 00113 | 001774 | B1774 | OCT | 1774 |
| 0113 | 00114 | 001775 | B1775 | OCT | 1775 |

357

PAGE 241

9830

| | | | | | |
|------|-------|--------|-------|-----|--------|
| 0114 | 00115 | 001777 | B1777 | OCT | 1777 |
| 0115 | 00116 | 002000 | QTOP | OCT | 2000 |
| 0116 | 00117 | 002400 | B2400 | OCT | 2400 |
| 0117 | 00120 | 003000 | B3000 | OCT | 3000 |
| 0118 | 00121 | 004000 | B4000 | OCT | 4000 |
| 0119 | 00122 | 006000 | B6K | OCT | 6000 |
| 0120 | 00004 | | B10K | EQU | C6+1 |
| 0121 | 00004 | | RPOP | EQU | B10K |
| 0122 | 00123 | 012000 | B12K | OCT | 12000 |
| 0123 | 00123 | | RBOP | EQU | B12K |
| 0124 | 00124 | 032000 | EXPOP | OCT | 32000 |
| 0125 | 00125 | 050000 | LBOP | OCT | 50000 |
| 0126 | 00126 | 052000 | LPOP | OCT | 52000 |
| 0127 | 00127 | 170000 | B170K | OCT | 170000 |
| 0128 | 00130 | 013777 | FROMA | OCT | 13777 |
| 0129 | 00131 | 000000 | LROMA | BSS | 1 |
| 0130 | 00132 | 177777 | M1 | DEC | -1 |
| 0131 | 00133 | 177776 | M2 | DEC | -2 |
| 0132 | 00134 | 177775 | M3 | DEC | -3 |
| 0133 | 00135 | 177774 | M4 | DEC | -4 |
| 0134 | 00136 | 177773 | M5 | DEC | -5 |
| 0135 | 00137 | 177772 | M6 | DEC | -6 |
| 0136 | 00140 | 177767 | M9 | DEC | -9 |
| 0137 | 00141 | 177766 | M10 | DEC | -10 |
| 0138 | 00142 | 177765 | M11 | DEC | -11 |
| 0139 | 00143 | 177764 | M12 | DEC | -12 |
| 0140 | 00144 | 177763 | M13 | DEC | -13 |
| 0141 | 00145 | 177762 | M14 | DEC | -14 |
| 0142 | 00146 | 177761 | M15 | DEC | -15 |
| 0143 | 00147 | 177751 | M23 | DEC | -23 |
| 0144 | 00150 | 177747 | M25 | DEC | -25 |
| 0145 | 00151 | 177740 | M32 | DEC | -32 |
| 0146 | 00152 | 177734 | M36 | DEC | -36 |
| 0147 | 00153 | 177720 | M48 | DEC | -48 |
| 0148 | 00154 | 177700 | M64 | DEC | -64 |
| 0149 | 00155 | 177660 | M80 | DEC | -80 |
| 0150 | 00156 | 177634 | M100 | DEC | -100 |
| 0151 | 00157 | 177400 | M256 | DEC | -256 |
| 0152 | 00160 | 176000 | M1024 | DEC | -1024 |
| 0153 | 00161 | 154360 | MAXSN | DEC | -10000 |

—(MAXIMUM ALLOWABLE LINE NUMBER)

0154 00162 043116 FN ASC 1,FN

0155*

MATH CONSTANTS THAT ARE NEEDED ON BASE PAGE

0156*

0157*

0158 00163 177401 ZAP OCT 177401

0159 00164 063440 MASK1 OCT 063440

0160 00165 061400 MASK5 OCT 061400

PAGE 242

9830

| | | | | | | | |
|-------|-------|--------|-------|-----|-------|--|---------------------------------|
| 0162* | | | | | | | |
| 0163 | 00166 | 000000 | FWAM | BSS | 1 | | FIRST WORD OF AVAILABLE MEM |
| 0164 | 00112 | | RSTKA | EQU | B1760 | | START OF RETURN STACK |
| 0165 | 00167 | 000000 | WRITE | BSS | 1 | | PRINTER AND DISPLAY DRIVER |
| 0166 | 00170 | 000000 | CLERA | BSS | 1 | | CLEAR SCREEN |
| 0167 | 00116 | | REED | EQU | QTOP | | KEYBOARD DRIVER QTOP IS = 2000B |
| 0168 | 00121 | | START | EQU | B4000 | | |
| 0169 | 00171 | 000000 | RUNA | BSS | 1 | | |
| 0170 | 00172 | 000000 | XEC4A | BSS | 1 | | |
| 0171 | 00173 | 000000 | PEXMA | BSS | 1 | | |
| 0172 | 00174 | 000000 | RDYDA | BSS | 1 | | |
| 0173 | 00122 | | LISTA | EQU | B6K | | |
| 0174 | 00175 | 000000 | TSRHA | BSS | 1 | | TABLE SEARCH |
| 0175 | 00176 | 000000 | ELINA | BSS | 1 | | OUTPUT I/O BUFFER |
| 0176 | 00177 | 000000 | SSYMA | BSS | 1 | | SEARCH SYMBOL TABLE FOR |
| | | | | | | | SYMBOL |
| 0177 | 00200 | 000000 | OPCHA | BSS | 1 | | |
| 0178 | 00201 | 000000 | FORMA | BSS | 1 | | |
| 0179 | 00202 | 000000 | FSCA | BSS | 1 | | |
| 0180 | 00203 | 000000 | ACTSA | BSS | 1 | | |
| 0181 | 00204 | 000000 | DELSA | BSS | 1 | | |
| 0182 | 00205 | 000000 | AERRA | BSS | 1 | | ALTERNATE ENTRY FOR ERROR |
| 0183 | 00206 | 000000 | ERRA | BSS | 1 | | |
| 0184 | 00207 | 014560 | INTEA | DEF | ISERV | | |
| 0185 | 00210 | 000000 | OUTIA | BSS | 1 | | OUTPUT AN INTEGER |
| 0186 | 00211 | 000000 | .BUFA | BSS | 1 | | FIRST CHAR OF I/O BUFFER |
| 0187 | 00212 | 000000 | .SLBF | BSS | 1 | | REFRESH BUFFER POINTER |
| 0188 | 00213 | 001744 | ARIA | DEF | AR1 | | |
| 0189 | 00214 | 001754 | AR2A | DEF | AR2 | | |
| 0190 | 00215 | 000000 | TMPOP | BSS | 1 | | |
| 0191 | 00216 | 000000 | .FADA | BSS | 1 | | ADD |
| 0192 | 00217 | 000000 | .FSRA | BSS | 1 | | SUBTRACT |
| 0193 | 00220 | 000000 | .FMPA | BSS | 1 | | MULTPY |
| 0194 | 00221 | 000000 | .FDVA | BSS | 1 | | DIVIDE |
| 0195 | 00222 | 000000 | ENTFA | BSS | 1 | | |
| 0196 | 00223 | 000000 | .FSR | BSS | 1 | | SQUARE ROOT |
| 0197 | 00224 | 000000 | UPARA | BSS | 1 | | UP ARROW |
| 0198 | 00225 | 000000 | FLTRA | BSS | 1 | | |
| 0199 | 00226 | 000000 | FLTFA | BSS | 1 | | |
| 0200 | 00227 | 000000 | FXFLA | BSS | 1 | | |
| 0201 | 00230 | 000000 | FLTSA | BSS | 1 | | |
| 0202 | 00231 | 000000 | SPLFA | BSS | 1 | | |
| 0203 | 00232 | 000000 | SNFLA | BSS | 1 | | |
| 0204 | 00233 | 000000 | BEND | BSS | 1 | | I/O BUFFER END |
| 0205 | 00233 | | SBUFA | EQU | BEND | | |
| 0206 | 00234 | 000000 | NMOTA | BSS | 1 | | NUMBER OUTPUT ROUTINE LINK |
| 0207 | 00235 | 000000 | FILLA | BSS | 1 | | SUPPLY NEEDED BLANKS ON |
| | | | | | | | OUTPUT |
| 0208 | 00236 | 000000 | SLEND | BSS | 1 | | REFRESH BUFFER END |
| 0209 | 00111 | | TMPPA | EQU | B1740 | | |
| 0210 | 00110 | | BADRP | EQU | B1400 | | POINTER POINTER TO I/O BUFFER |
| 0211* | | | | | | | |
| 0212 | 00123 | | FOPBS | EQU | B12K | | |
| 0213 | 00237 | 000000 | SALTA | BSS | 1 | | |
| 0214 | 00240 | 000000 | MDIMA | BSS | 1 | | |
| 0215 | 00241 | 000000 | SYTSA | BSS | 1 | | |
| 0216 | 00242 | 000000 | DCOMA | BSS | 1 | | |
| 0217 | 00243 | 000000 | NUMCA | BSS | 1 | | |

389

347

1,444,141

347

PAGE 243

9830

| | | | | | |
|-------|-------|--------|-------|-----|---|
| 0218 | 00244 | 000000 | NUMOA | BSS | 1 |
| 0219 | 00245 | 000000 | GTSTA | BSS | 1 |
| 0220 | 00246 | 000000 | VAROA | BSS | 1 |
| 0221 | 00247 | 000000 | ECALA | BSS | 1 |
| 0222 | 00250 | 000000 | OVCHA | BSS | 1 |
| 0223 | 00251 | 000000 | CLPIA | BSS | 1 |
| 0224 | 00252 | 000000 | ENTEA | BSS | 1 |
| 0225 | 00253 | 000000 | CHRSA | BSS | 1 |
| 0226 | 00254 | 000000 | XSCRA | BSS | 1 |
| 0227 | 00255 | 000000 | XEC6A | BSS | 1 |
| 0228 | 00256 | 000000 | FDAAA | BSS | 1 |
| 0229 | 00257 | 000000 | FNDAA | BSS | 1 |
| 0230 | 00260 | 000000 | FNDIA | BSS | 1 |
| 0231 | 00261 | 000000 | IOSTA | BSS | 1 |
| 0232 | 00262 | 000000 | INREA | BSS | 1 |
| 0233 | 00263 | 000000 | SUMPA | BSS | 1 |
| 0234 | 00264 | 000000 | CRCKA | BSS | 1 |
| 0235 | 00265 | 000000 | IOHEA | BSS | 1 |
| 0236 | 00266 | 000000 | QTCHA | BSS | 1 |
| 0237 | ***** | | | | |
| 0238* | | | | | |
| 0239 | 00267 | 000000 | | BSS | 1 |

CHECKSUM WORD*****

PAGE 244

9380—ERROR TABLE

| | | | | | |
|------|-------|--------|-----|-----|---|
| 0241 | 00270 | 000000 | E1 | BSS | 1 |
| 0242 | 00271 | 000000 | | BSS | 1 |
| 0243 | 00272 | 000000 | | BSS | 1 |
| 0244 | 00273 | 000000 | | BSS | 1 |
| 0245 | 00274 | 000000 | E5 | BSS | 1 |
| 0246 | 00275 | 000000 | | BSS | 1 |
| 0247 | 00276 | 000000 | | BSS | 1 |
| 0248 | 00277 | 000000 | | BSS | 1 |
| 0249 | 00300 | 000000 | | BSS | 1 |
| 0250 | 00301 | 000000 | E10 | BSS | 1 |
| 0251 | 00302 | 000000 | | BSS | 1 |
| 0252 | 00303 | 000000 | | BSS | 1 |
| 0253 | 00304 | 000000 | | BSS | 1 |
| 0254 | 00305 | 000000 | | BSS | 1 |
| 0255 | 00306 | 000000 | E15 | BSS | 1 |
| 0256 | 00307 | 000000 | | BSS | 1 |
| 0257 | 00310 | 000000 | | BSS | 1 |
| 0258 | 00311 | 000000 | | BSS | 1 |
| 0259 | 00312 | 000000 | | BSS | 1 |
| 0260 | 00313 | 000000 | E20 | BSS | 1 |
| 0261 | 00314 | 000000 | | BSS | 1 |
| 0262 | 00315 | 000000 | | BSS | 1 |
| 0263 | 00316 | 000000 | | BSS | 1 |
| 0264 | 00317 | 000000 | | BSS | 1 |
| 0265 | 00320 | 000000 | E25 | BSS | 1 |
| 0266 | 00321 | 000000 | | BSS | 1 |
| 0267 | 00322 | 000000 | | BSS | 1 |
| 0268 | 00323 | 000000 | | BSS | 1 |
| 0269 | 00324 | 000000 | | BSS | 1 |
| 0270 | 00325 | 000000 | E30 | BSS | 1 |
| 0271 | 00326 | 000000 | | BSS | 1 |
| 0272 | 00327 | 000000 | | BSS | 1 |
| 0273 | 00330 | 000000 | | BSS | 1 |
| 0274 | 00331 | 000000 | | BSS | 1 |
| 0275 | 00332 | 000000 | E35 | BSS | 1 |
| 0276 | 00333 | 000000 | | BSS | 1 |
| 0277 | 00334 | 000000 | | BSS | 1 |
| 0278 | 00335 | 000000 | | BSS | 1 |
| 0279 | 00336 | 000000 | | BSS | 1 |
| 0280 | 00337 | 000000 | E40 | BSS | 1 |
| 0281 | 00340 | 000000 | | BSS | 1 |
| 0282 | 00341 | 000000 | | BSS | 1 |
| 0283 | 00342 | 000000 | | BSS | 1 |
| 0284 | 00343 | 000000 | | BSS | 1 |
| 0285 | 00344 | 000000 | E45 | BSS | 1 |
| 0286 | 00345 | 000000 | | BSS | 1 |
| 0287 | 00346 | 000000 | | BSS | 1 |
| 0288 | 00347 | 000000 | | BSS | 1 |
| 0289 | 00350 | 000000 | | BSS | 1 |
| 0290 | 00351 | 000000 | E50 | BSS | 1 |
| 0291 | 00352 | 000000 | | BSS | 1 |
| 0292 | 00353 | 000000 | | BSS | 1 |
| 0293 | 00354 | 000000 | | BSS | 1 |
| 0294 | 00355 | 000000 | | BSS | 1 |

PAGE 245

9830—BASE PAGE READ-WRITE MEMORY

| | | | | |
|-------|-------------|--------|-------------|----------------------------------|
| 0296 | 01400 | | ORG 1400B | |
| 0297 | 01400 | 000000 | BSS 1 | CURRENT CHAR POINTER |
| 0298 | 01401 | 000000 | BSS 1 | SAVE (A) FOR RETN 2 |
| 0299 | 01402 | 000000 | BSS 1 | REFRESH POINTER |
| 0300 | 01403 | 000000 | BSS 1 | LAST WORD OF R/W MEM |
| 0301 | 01404 | 000000 | BSS 1 | LAST WORD AVAILABLE FOR USER |
| 0302 | 01405 | 000000 | BSS 1 | FIRST WORD OF COMMON OR MAINLINE |
| 0303 | 01406 | 000000 | BSS 1 | FIRST WORD -1 OF VALUE TABLE |
| 0304 | 01407 | 000000 | BSS 1 | FIRST WORD -1 OF SYMBOL TABLE |
| 0305 | 01410 | 000000 | BSS 1 | FIRST WORD OF OPERATOR STACK |
| 0306 | 01411 | 000000 | BSS 1 | TEMPORARY STACK POINTER |
| 0307 | 01412 | 000000 | BSS 1 | OPERATOR STACK POINTER |
| 0308 | 01413 | 000000 | BSS 1 | OPERAND STACK POINTER |
| 0309 | 01414 | 000000 | BSS 1 | DATA STATEMENT |
| 0310 | 01415 | 000000 | BSS 1 | POINTERS |
| 0311 | 01416 | 000000 | BSS 1 | PROGRAM=0; KEYBOARD=1 |
| 0312 | 01417 | 000000 | BSS 1 | PARAMETER ADDRESS |
| 0313 | 01420 | 000000 | BSS 1 | PARAMETER NAME |
| 0314 | 01421 | 000000 | BSS 1 | LAST WORD+1 OF CURRENT STMT |
| 0315 | 01422 | 000000 | BSS 1 | USER KEY POINTER |
| 0316 | 01423 | 000000 | BSS 1 | LAST WORD OF CURRENT PROGRAM |
| 0317 | 01424 | 000000 | BSS 1 | FIRST WORD OF CURRENT PROGRAM |
| 0318 | 01425 | 000000 | BSS 1 | FIRST WORD OF MAINLINE |
| 0319 | 01426 | 000000 | BSS 1 | LAST WORD+1 OF MAINLINE |
| 0320 | 01427 | 000000 | BSS 1 | CURRENT LINE NUMBER |
| 0321 | 01430 | 000000 | BSS 1 | AUTO LINE INCREMENT |
| 0322 | 01431 | 000000 | BSS 1 | CURRENT LINE FOR LIST |
| 0323 | 01432 | 000000 | BSS 1 | INPUT STMT FLAG |
| 0324 | 01433 | 000000 | BSS 1 | PTAPE FLAG |
| 0325 | 01434 | 000000 | BSS 1 | TRIG UNITS |
| 0326 | 01435 | 000000 | BSS 1 | SECURE PROGRAM FLAG |
| 0327 | 01436 | 000000 | BSS 2 | STOP ADDRESS |
| 0328 | 01440 | 000000 | BSS 2 | TRACE LIMIT ADDRESSES |
| 0329 | 01442 | 000000 | BSS 1 | STOP FLAG |
| 0330 | 01443 | 000000 | BSS 1 | KEYBOARD FLAG |
| 0331 | 01444 | 000000 | BSS 1 | INTERRUPT |
| 0332 | 01445 | 000000 | BSS 1 | |
| 0333 | 01446 | 000000 | BSS 1 | TEMPORARIES |
| 0334 | 01447 | 000000 | BSS 1 | SELECT CODE |
| 0335 | 01450 | 000000 | BSS 1 | PRINT-ALL FLAG |
| 0336 | 01451 | 000000 | BSS 1 | KEYBOARD |
| 0337 | 01452 | 000000 | BSS 1 | FORMAT FLAGS |
| 0338 | 01453 | 000000 | BSS 1 | PRINT/DISP FLAG |
| 0339 | 01454 | 000000 | BSS 1 | FORMAT |
| 0340 | 01455 | 000000 | BSS 1 | |
| 0341 | 01456 | 000000 | BSS 1 | |
| 0342 | 01457 | 000000 | BSS 1 | STATEMENT FLAGS |
| 0343 | 01460 | 000000 | BSS 1 | CURRENT OPERATOR CODE |
| 0344 | 01461 | 000000 | BSS 1 | CURRENT OPERATOR PRECEDENCE |
| 0345 | 01462 | 000000 | BSS 1 | |
| 0346 | 01463 | 000000 | BSS 1 | |
| 0347 | 01464 | 000000 | BSS 2 | UNUSED |
| 0348* | | | | |
| 0349* | TEMPORARIES | | | |
| 0350* | | | | |
| 0351 | 01466 | 000000 | LOCL1 BSS 1 | |

PAGE 246

9830—BASE PAGE READ-WRITE MEMORY

| | | | | | | |
|-------|--|--------|-------|-----|------|--|
| 0352 | 01467 | 000000 | LOCL2 | BSS | 1 | |
| 0353 | 01470 | 000000 | L2T1 | BSS | 1 | |
| 0354 | 01471 | 000000 | L2T2 | BSS | 1 | |
| 0355 | 01472 | 000000 | TEMPS | BSS | 1 | |
| 0356 | 01473 | 000000 | TEMP | BSS | 1 | |
| 0357 | 01473 | | MBOX1 | EQU | TEMP | |
| 0358 | 01474 | 000000 | TEMP1 | BSS | 1 | |
| 0359 | 01475 | 000000 | TEMP2 | BSS | 1 | |
| 0360 | 01476 | 000000 | TEMP3 | BSS | 1 | |
| 0361 | 01477 | 000000 | TEMP4 | BSS | 1 | |
| 0362* | | | | | | |
| 0363* | BUFFERS | | | | | |
| 0364* | | | | | | |
| 0365 | 01500 | 000000 | IOBUF | BSS | 41 | |
| 0366 | 01551 | 000000 | RBUF | BSS | 41 | INPUT AND RECALL BUFFER
SINGLE LINE RESEARCH BUFFER |
| 0367* | | | | | | |
| 0368* | PRINT ROUTINE FLAGS | | | | | |
| 0369* | | | | | | |
| 0370 | 01622 | 000000 | TERM | BSS | 1 | |
| 0371 | 01623 | 000000 | PEND | BSS | 1 | |
| 0372 | 01624 | 000000 | PPFF | BSS | 1 | |
| 0373 | 01625 | 000000 | FIRST | BSS | 1 | |
| 0374 | 01626 | 000000 | | BSS | 2 | |
| 0375* | | | | | | UNUSED |
| 0376* | MATH TEMPORARIES | | | | | |
| 0377* | | | | | | |
| 0378 | 01630 | 000000 | TREGE | BSS | 4 | |
| 0379 | 01634 | 000000 | RSLTE | BSS | 4 | |
| 0380* | | | | | | |
| 0381* | STATEMENT SYNTAX AND EXECUTION TEMPORARIES | | | | | |
| 0382* | | | | | | |
| 0383 | 01640 | 000000 | ST | BSS | 32 | |
| 0384* | | | | | | |
| 0385* | SYSTEM SYNTAX AND EXECUTION TEMPORARIES | | | | | |
| 0386* | | | | | | |
| 0387 | 01700 | 000000 | | BSS | 1 | |
| 0388 | 01701 | 000000 | NDISP | BSS | 1 | |
| 0389 | 01702 | 000000 | NPRIN | BSS | 1 | |
| 0390 | 01703 | 000000 | PLACE | BSS | 1 | |
| 0391 | 01704 | 000000 | WIDTH | BSS | 1 | |
| 0392 | 01705 | 000000 | FLOAT | BSS | 1 | |
| 0393 | 01706 | 000000 | SBPTR | BSS | 1 | |
| 0394 | 01707 | 000000 | BLANK | BSS | 1 | |
| 0395 | 01710 | 000000 | MSFLG | BSS | 1 | |
| 0396 | 01711 | 000000 | DFLAG | BSS | 1 | |
| 0397 | 01712 | 000000 | PFLAG | BSS | 1 | |
| 0398 | 01713 | 000000 | UFLAG | BSS | 1 | |
| 0399 | 01714 | 000000 | SFLAG | BSS | 1 | |
| 0400 | 01715 | 000000 | SBADR | BSS | 1 | |
| 0401 | 01716 | 000000 | SSBPT | BSS | 1 | |
| 0402 | 01717 | 000000 | NUM | BSS | 1 | |
| 0403 | 01717 | | OPTYP | EQU | NUM | |
| 0404 | 01720 | 000000 | LASTN | BSS | 1 | |
| 0405 | 01721 | 000000 | SIGN | BSS | 1 | |
| 0406 | 01721 | | MAXIN | EQU | SIGN | |
| 0407 | 01722 | 000000 | EXP | BSS | 1 | |

SAVED FOR PROCESSOR

PAGE 247

9830 — BASE PAGE READ-WRITE MEMORY

| | | | | | |
|------|-------|--------|-------|-----|-----|
| 0408 | 01722 | | INTGR | EQU | EXP |
| 0409 | 01723 | 000000 | ARYAD | BSS | 1 |
| 0410 | 01724 | 000000 | GFLAG | BSS | 1 |
| 0411 | 01725 | 000000 | NXTST | BSS | 1 |
| 0412 | 01726 | 000000 | MBIN1 | BSS | 1 |
| 0413 | 01727 | 000000 | MBIN2 | BSS | 1 |

0414*

0415*

ARITHMETIC REGISTERS AND RETURN STACK

0416*

| | | | | | |
|------|-------|--------|-------|-----|-------|
| 0417 | 01740 | | | ORG | 1740B |
| 0418 | 01740 | 000000 | XC | BSS | 4 |
| 0419 | 01744 | 000000 | AR1 | BSS | 4 |
| 0420 | 01750 | 000000 | YC | BSS | 4 |
| 0421 | 01754 | 000000 | AR2 | BSS | 4 |
| 0422 | 01760 | 000000 | RSTAK | BSS | 11 |
| 0423 | 01773 | 000000 | RES | BSS | 4 |
| 0424 | 01777 | 000000 | RSPTR | BSS | 1 |
| 0425 | 00356 | | RETN3 | EQU | 356B |
| 0426 | 00357 | | RETN2 | EQU | 357B |
| 0427 | 00365 | | RET2 | EQU | 365B |
| 0428 | 00373 | | ICNT | EQU | 373B |
| 0429 | 00374 | | WBUFF | EQU | 374B |
| 0430 | 00400 | | GPARM | EQU | 400B |
| 0431 | 00405 | | XFAR1 | EQU | 405B |
| 0432 | 00412 | | RSULT | EQU | 412B |
| 0433 | 00416 | | STORE | EQU | 416B |
| 0434 | 00425 | | STOR1 | EQU | 425B |
| 0435 | 00444 | | XFAR2 | EQU | 444B |
| 0436 | 00450 | | STROP | EQU | 450B |
| 0437 | 00456 | | SBPUD | EQU | 456B |
| 0438 | 00461 | | SYMC2 | EQU | 461B |
| 0439 | 00463 | | SYMC1 | EQU | 463B |
| 0440 | 00464 | | SYMCK | EQU | 464B |
| 0441 | 00504 | | COMCK | EQU | 504B |
| 0442 | 00510 | | COMCE | EQU | 510B |
| 0443 | 00513 | | LPCK | EQU | 513B |
| 0444 | 00521 | | LPCKE | EQU | 521B |
| 0445 | 00524 | | RPCK | EQU | 524B |
| 0446 | 00534 | | RPCKE | EQU | 534B |
| 0447 | 00535 | | FSCE2 | EQU | 535B |
| 0448 | 00537 | | STSRH | EQU | 537B |
| 0449 | 00546 | | STEXP | EQU | 546B |
| 0450 | 00553 | | BHSTP | EQU | 553B |
| 0451 | 00561 | | MER8 | EQU | 561B |
| 0452 | 00561 | | FSCE4 | EQU | 561B |
| 0453 | 00563 | | SLWST | EQU | 563B |
| 0454 | 00567 | | UNSTK | EQU | 567B |
| 0455 | 00574 | | NEXTL | EQU | 574B |
| 0456 | 00606 | | NEXT1 | EQU | 606B |
| 0457 | 00613 | | FPAGE | EQU | 613B |
| 0458 | 00626 | | SAVBP | EQU | 626B |
| 0459 | 00633 | | RESB1 | EQU | 633B |
| 0460 | 00640 | | RESBP | EQU | 640B |
| 0461 | 00645 | | MINIT | EQU | 645B |
| 0462 | 00651 | | INITS | EQU | 651B |
| 0463 | 00664 | | LNUM | EQU | 664B |

RETURN STACK
RESULT REGISTER
RETURN STACK POINTER

PAGE 248

9830 — BASE PAGE READ-WRITE MEMORY

| | | | | |
|------|-------|-------|-----|-------|
| 0464 | 00666 | PGINT | EQU | 666B |
| 0465 | 00676 | INTCK | EQU | 676B |
| 0466 | 00726 | SYE25 | EQU | 726B |
| 0467 | 00730 | GETAD | EQU | 730B |
| 0468 | 00742 | FND | EQU | 742B |
| 0469 | 00750 | DIGCK | EQU | 750B |
| 0470 | 00752 | LETCK | EQU | 752B |
| 0471 | 00753 | LIMIT | EQU | 753B |
| 0472 | 00771 | GTOPP | EQU | 771B |
| 0473 | 01007 | NEXTA | EQU | 1007B |
| 0474 | 01015 | OBLNK | EQU | 1015B |
| 0475 | 01016 | ONEXT | EQU | 1016B |
| 0476 | 01017 | OUTCR | EQU | 1017B |
| 0477 | 01041 | GETCH | EQU | 1041B |
| 0478 | 01053 | GTMP | EQU | 1053B |
| 0479 | 01055 | GNEXT | EQU | 1055B |
| 0480 | 01062 | GFRST | EQU | 1062B |
| 0481 | 01066 | GETSK | EQU | 1066B |
| 0482 | 01075 | FNDPS | EQU | 1075B |
| 0483 | 01121 | MVTOH | EQU | 1121B |
| 0484 | 01131 | PGIN0 | EQU | 1131B |
| 0485 | 01140 | LTR | EQU | 1140B |
| 0486 | 01147 | ENDS | EQU | 1147B |
| 0487 | 01151 | EOST | EQU | 1151B |
| 0488 | 01152 | NOEOF | EQU | 1152B |
| 0489 | 01153 | EOLCK | EQU | 1153B |
| 0490 | 01156 | CONST | EQU | 1156B |
| 0491 | 01165 | SYEI2 | EQU | 1165B |
| 0492 | 01166 | GSIGN | EQU | 1166B |
| 0493 | 01201 | IMPY | EQU | 1201B |
| 0494 | 01217 | FETCH | EQU | 1217B |
| 0495 | 01225 | LDEF | EQU | 1225B |
| 0496 | 01254 | ZONK | EQU | 1254B |
| 0497 | 01256 | CLAR2 | EQU | 1256B |
| 0498 | 01314 | ODGIT | EQU | 1314B |
| 0499 | 01317 | SGNS | EQU | 1317B |
| 0500 | 01330 | .NPWR | EQU | 1330B |
| 0501 | 01335 | NPWRT | EQU | 1335B |
| 0502 | 01337 | NDWRT | EQU | 1337B |
| 0503 | 01342 | CRLF | EQU | 1342B |
| 0504 | 01344 | CRLF1 | EQU | 1344B |
| 0505 | 01351 | TTY | EQU | 1351B |
| 0506 | 01362 | GTCON | EQU | 1362B |
| 0507 | 01371 | FIX | EQU | 1371B |

PAGE 249

9830A TAPE CASSETTE SYSTEM

0510 LST
 0511* CTAPE 3-20-72-9:30
 0512 14000 ORG 14000B
 0513*
 0514* SYSTEM COMMAND TABLE
 0515*
 0516 14000 000003 ABS ESEC-*
 0517 14001 051505 CMNDT ASC 1,SE SECURE, OP CODE 1
 0518 14002 041761 OCT 41761 C, OP CODE
 0519 14003 060730 ESEC JSM GETAD A=LNK1,B=LNK2
 0520 14004 035647 STB TAPI TEMPOR; LNK2 ADDR
 0521 14005 025477 LDB TEMP4 LNK1 ADDR
 0522 14006 015647 ESEC3 CPB TAPI DONE?
 0523 14007 066016 JMP ESEC6
 0524 14010 074517 LDA B,I
 0525 14011 070410 SZA ESEC4 SKIP IF TYPING AID
 0526 14012 072053 SAP *+1,S SET BIT 15
 0527 14013 074557 STA B,I RESTORE LINE
 0528 14014 061007 JSM NEXTA NO, GET NEXT LINE
 0529 14015 066006 JMP ESEC3
 0530 14016 021435 ESEC6 LDA SECFL
 0531 14017 040062 IOR FLGBT SET BIT 15
 0532 14020 031435 STA SECFL CUSTOMER SECURING PROG
 0533 14021 164174 ESEC4 JMP RDYDA,I ET TO SYSTEM
 0534*
 0535*
 0536 14022 000000 BSS 1 ***** C.S. *****
 0537*
 0538*
 0539* SUB-TABLE FOR STORE AND LOAD
 0540*
 0541*
 0542 14023 041111 SUBT ASC 1,BI BIN OP CODE 24
 0543 14024 047264 OCT 47264 N, OP CODE
 0544 14025 045505 ASC 1,KE KEY, OP CODE 20
 0545 14026 054660 OCT 54660 Y, OP CODE
 0546 14027 042101 ASC 2,DATA DATA, OP CODE 21
 0547 14030 052101
 0548* 14031 170400 OCT 170400 OP CODE, EOT
 0549*
 0550* SYNTAX SERVICE SUBR
 0551*
 0552*
 0553 14032 061055 SERV JSM GNEXT
 0554 14033 060456 JSM SBPUD
 0555 14034 010031 CPA EOL EOL?
 0556 14035 170402 RET
 0557 14036 055706 DSZ SBPTR
 0558 14037 055400 DSZ BADDR
 0559 14040 160202 JSM FSCL,I GEET NO. OR EXPRESSION
 0560 14041 064357 JMP RETN2
 0561*
 0562*
 0563*
 0564*

354

354

1,444,141

354

PAGE 250

9830A TAPE CASSETTE SYSTEM

| | | | | | |
|-------|-------|--------|-------|-----------|---------|
| 0565* | | | | EXECUTION | |
| 0566* | | | | | |
| 0567* | | | | | |
| 0568* | | | | | |
| 0569* | | | | | |
| 0570* | | | | | |
| 0571* | | | | | |
| 0572* | | | | | |
| 0573 | 14042 | 060771 | ALOAD | JSM | GTOPP |
| 0574 | 14043 | 072510 | | RZA | LOADP |
| 0575 | 14044 | 045472 | | ISZ | TEMPS |
| 0576 | 14045 | 121472 | | LDA | TEMPS,I |
| 0577 | 14046 | 013732 | | CPA | DATOP |
| 0578 | 14047 | 067172 | | JMP | LDATA |
| 0579 | 14050 | 013733 | | CPA | KEYOP |
| 0580 | 14051 | 067315 | | JMP | LKEY |
| 0581 | 14052 | 013734 | | CPA | BINOP |
| 0582 | 14053 | 067421 | | JMP | LBIN |
| 0583 | 14054 | 055472 | | DSZ | TEMPS |
| 0584* | | | | | |
| 0585* | | | | | |
| 0586* | | | | | |
| 0587* | | | | | |
| 0588* | | | | | |
| 0589 | 14055 | 062070 | LOADP | JSM | LOAD2 |
| 0590 | 14056 | 062102 | | JSM | LOAD8 |
| 0591 | 14057 | 060645 | | JSM | MINIT |
| 0592 | 14060 | 021660 | | LDA | TAP7 |
| 0593 | 14061 | 062160 | | JSM | LOAD3 |
| 0594 | 14062 | 021416 | LOAD9 | LDA | MODE |
| 0595 | 14063 | 070150 | | SZA | *+3 |
| 0596 | 14064 | 060651 | | JSM | INITS |
| 0597 | 14065 | 164174 | | JMP | RDYDA,I |
| 0598 | 14066 | 021661 | | LDA | TAP8 |
| 0599 | 14067 | 164171 | | JMP | RUNA,I |
| 0600* | | | | | |
| 0601* | | | | | |
| 0602* | | | | | |
| 0603 | 14070 | 062300 | LOAD2 | JSM | SFLOC |
| 0604 | 14071 | 027745 | | LDB | TAPAD |
| 0605 | 14072 | 021472 | LO3 | LDA | TEMPS |
| 0606 | 14073 | 011421 | | CPA | PRADD |
| 0607 | 14074 | 170402 | | RET | |
| 0608 | 14075 | 045472 | | ISZ | TEMPS |
| 0609 | 14076 | 121472 | | LDA | TEMPS,I |
| 0610 | 14077 | 074557 | | STA | B,I |
| 0611 | 14100 | 045472 | | ISZ | TEMPS |
| 0612 | 14101 | 077470 | | RIB | LO3 |
| 0613* | | | | | |
| 0614* | | | | | |
| 0615* | | | | | |
| 0616 | 14102 | 021661 | LOAD8 | LDA | TAP8 |
| 0617 | 14103 | 070110 | | SZA | *+2 |
| 0618 | 14104 | 066126 | | JMP | LOAD7-2 |
| 0619 | 14105 | 021416 | | LDA | MODE |
| 0620 | 14106 | 070110 | | SZA | *+2 |

ALOAD — ALL LOAD EXECT ENTRY

LOADP — LOAD PROGRAM

SRATCH SYMBOL TABLE

RESET STACKS
KEYB-RET TO SYSTEM-DO NOT
EXECUTE
SEC LINE #

TEMPOR, ADDR
STATEMENT POINTER
EOL FOLLOWS?
YES, PROCESS
POINT TO LINE #
LINE #
TAP-TAP8, LINE #S
LOOK FOR SECOND LINE #

LNx2 GIVEN
SKIP ON PROGRAM

367

PAGE 251

9830A TAPE CASSETTE SYSTEM

| | | | | | | |
|-------|-------|--------|-------|-----|---------|-----------------------------------|
| 0621 | 14107 | 066130 | | JMP | LOAD7 | DO NOT EXECUTE |
| 0622 | 14110 | 121421 | | LDA | PRADD,I | GET NEXT LINE # |
| 0623 | 14111 | 070072 | | SAM | *+1,C | CLEAR SEC BIT |
| 0624 | 14112 | 025421 | | LDB | PRADD | |
| 0625 | 14113 | 015423 | | CPB | PBPTR | |
| 0626 | 14114 | 020056 | | LDA | .1E4 | 10000 |
| 0627 | 14115 | 070076 | | TCA | | |
| 0628 | 14116 | 001660 | | ADA | TAP7 | LNK1 — LINE # FOR START LOAD |
| 0629 | 14117 | 070213 | | SAP | *+4 | NEXT LINE OK |
| 0630 | 14120 | 021660 | | LDA | TAP7 | START AT FIRST LINE OF LOADED |
| | | | | | | PROG. |
| 0631 | 14121 | 031661 | | STA | TAP8 | FOR START OF EXECUTION |
| 0632 | 14122 | 066126 | | JMP | *+4 | |
| 0633 | 14123 | 021427 | | LDA | .LNUM | PRESENT L.N. |
| 0634 | 14124 | 070070 | | SIA | *+1 | |
| 0635 | 14125 | 031661 | | STA | TAP8 | START AT NEXT N←N←LINE |
| 0636 | 14126 | 070742 | | SAR | 16 | |
| 0637 | 14127 | 031416 | | STA | MODE | EXECUTE |
| 0638 | 14130 | 025421 | LOAD7 | LDB | PRADD | |
| 0639 | 14131 | 163756 | | JSM | DCM1A,I | DECOMPILE |
| 0640 | 14132 | 021666 | | LDA | IDB0+2 | FILE TYPE |
| 0641 | 14133 | 010020 | | CPA | .3 | SOURCE? |
| 0642 | 14134 | 066142 | | JMP | *+6 | |
| 0643 | 14135 | 010036 | | CPA | .23 | SEC SOURCE ? |
| 0644 | 14136 | 066140 | | JMP | *+2 | YES |
| 0645 | 14137 | 067325 | | JMP | ERR7 | TYPE ERROR |
| 0646 | 14140 | 024106 | | LDB | B777 | |
| 0647 | 14141 | 035435 | | STB | SECFL | SET SECURE FLAG |
| 0648 | 14142 | 021424 | | LDA | PBUFF | |
| 0649 | 14143 | 011423 | | CPA | PBPTR | NULL PROG? |
| 0650 | 14144 | 066151 | | JMP | LO12 | YES |
| 0651 | 14145 | 121424 | | LDA | PBUFF,I | TYPING AID RESIDENT? |
| 0652 | 14146 | 072150 | | RZA | LO12 | |
| 0653 | 14147 | 160205 | ERR8 | JSM | AERRA,I | OVERLAY ERROR |
| 0654 | 14150 | 177701 | | DEC | -63 | |
| 0655 | 14151 | 025660 | LO12 | LDB | TAP7 | |
| 0656 | 14152 | 074210 | | SZB | *+4 | START LINE SPECIFIED ? |
| 0657 | 14153 | 025670 | | LDB | IDB0+4 | YES, COMPUTE DIFF |
| 0658 | 14154 | 074076 | | TCB | | BETWEEN SPECIFIED |
| 0659 | 14155 | 005660 | | ADB | TAP7 | AND ACTUAL LINE # |
| 0660 | 14156 | 035663 | | STB | OFFSET | TEMPOR |
| 0661 | 14157 | 170402 | | RET | | |
| 0662* | | | | | | |
| 0663* | | | | | | |
| 0664* | | | | | | |
| 0665 | 14160 | 061075 | LOAD3 | JSM | FNDPS | FIND ADDR OF START LINE |
| 0666 | 14161 | 066271 | | JMP | LO6 | L.N. BEYOND END OF PROGR |
| 0667 | 14162 | 066163 | | JMP | *+1 | L.N. BETWEEN PROGRAM LINES |
| 0668 | 14163 | 035650 | | STB | TAP2 | L. N. EXACTLY FOUND — STORE START |
| | | | | | | ADDR |
| 0669 | 14164 | 035477 | | STB | TEMP4 | SAVE FOR CLPIA |
| 0670 | 14165 | 025423 | | LDB | PBPTR | |
| 0671 | 14166 | 160251 | | JSM | CLPIA,I | DEL FROM HERE TO END OF |
| | | | | | | PROGR |
| 0672* | | | | | | |
| 0673* | | | | | | |
| 0674* | | | | | | |
| 0675 | 14167 | 021671 | LOAD4 | LDA | IDB0+5 | END LINE # |
| 0676 | 14170 | 001663 | | ADA | OFFSET | |

PAGE 252

9830A TAPE CASSETTE SYSTEM

| | | | | | |
|-------|-------|--------|-------|---------|----------------------------------|
| 0677 | 14171 | 000012 | ADA | .1 | |
| 0678 | 14172 | 060742 | JSM | FND | FIND ADDR OF END LINE |
| 0679 | 14173 | 015650 | CPB | TAP2 | SAVE AS START ADDR |
| 0680 | 14174 | 066176 | JMP | *+2 | YES |
| 0681 | 14175 | 066147 | JMP | ERR8 | NO, OVERLAY |
| 0682 | 14176 | 035660 | STB | TAP7 | TEMPOR |
| 0683* | | | | | |
| 0684* | | | | | |
| 0685* | | | | | |
| 0686* | | | | | |
| 0687 | 14177 | 021672 | COMON | LDA | IDB0+6 |
| 0688 | 14200 | 072210 | | RZA | YCOMN |
| 0689 | 14201 | 021665 | | LDA | IDB0+1 |
| 0690 | 14202 | 160250 | | JSM | OVCHA,I |
| 0691 | 14203 | 066231 | | JMP | KCOMN |
| 0692 | 14204 | 025660 | YCOMN | LDB | TAP7 |
| 0693 | 14205 | 015424 | | CPB | PBUFF |
| 0694 | 14206 | 066210 | | JMP | *+2 |
| 0695 | 14207 | 164330 | | JMP | E30+3,I |
| 0696 | 14210 | 025422 | | LDB | UKPTR |
| 0697 | 14211 | 076350 | | RZB | COM1 |
| 0698 | 14212 | 025405 | | LDB | FWUP |
| 0699 | 14213 | 035424 | | STB | PBUFF |
| 0700 | 14214 | 035426 | | STB | PBPTO |
| 0701 | 14215 | 035423 | | STB | PBPTR |
| 0702 | 14216 | 035476 | | STB | TEMP3 |
| 0703 | 14217 | 066221 | | JMP | *+2 |
| 0704 | 14220 | 070742 | COM1 | SAR | 16 |
| 0705 | 14221 | 001665 | ADA | IDB0+1 | DO NOT ALLOCATE SPACE FOR COMMON |
| 0706 | 14222 | 160250 | JSM | OVCHA,I | ADD PROGR SIZE TO COMMON SIZE |
| 0707 | 14223 | 021422 | | LDA | UKPTR |
| 0708 | 14224 | 072250 | | RZA | KCOMN |
| 0709 | 14225 | 021424 | | LDA | PBUFF |
| 0710 | 14226 | 001672 | ADA | IDB0+6 | SKIP IF KEY |
| 0711 | 14227 | 031424 | STA | PBUFF | UPDATE PBUFF |
| 0712 | 14230 | 031660 | STA | TAP7 | UPDATE LOAD ADDR |
| 0713 | 14231 | 061121 | KCOMN | JSM | MVTOH |
| 0714* | | | | | OPEN UP SPACE |
| 0715* | | | | | |
| 0716* | | | | | |
| 0717 | 14232 | 063105 | JSM | RRID | REV READ BOF; READ IMMDET FILE |
| 0718 | 14233 | 021660 | LDA | TAP7 | START ADDR |
| 0719 | 14234 | 025665 | LDB | IDB0+1 | CURR FILE SIZE |
| 0720 | 14235 | 070037 | ADB | A | END ADDR |
| 0721 | 14236 | 062717 | JSM | TRD | READ FILE |
| 0722 | 14237 | 063031 | JSM | HALT | |
| 0723 | 14240 | 070115 | SEC | *+2 | |
| 0724 | 14241 | 067100 | JMP | ERR4 | CHECK SUM ERROR |
| 0725* | | | | | |
| 0726* | | | | | |
| 0727* | | | | | |
| 0728 | 14242 | 021423 | LDA | PBPTR | |
| 0729 | 14243 | 031647 | STA | TAP1 | SAVE END OF PROG POINTER |
| 0730 | 14244 | 021654 | LDA | TAP5 | |
| 0731 | 14245 | 031423 | STA | PBPTR | SET TO END OF LOADED AREA |
| 0732 | 14246 | 021653 | LDA | TAP4 | POINTER TO START OF LOADED AREA |

369

| | | | | | |
|-------|-------|--------|-------|---------|-------------------------------|
| 0733 | 14247 | 160257 | JSM | FNDAA,I | FIND FIRST LINE /# REF |
| 0734 | 14250 | 066256 | JMP | LO4 | |
| 0735 | 14251 | 001663 | ADA | OFFSET | ADD OFFSET |
| 0736 | 14252 | 131472 | STA | TEMPS,I | RESTORE |
| 0737 | 14253 | 160260 | JSM | FNDIA,I | FIND NEXT REF |
| 0738 | 14254 | 066256 | JMP | LO4 | END OF LOADED AREA |
| 0739 | 14255 | 066251 | JMP | *-4 | |
| 0740* | | | | | |
| 0741* | | | | | |
| 0742* | | | | | |
| 0743* | | | | | |
| 0744 | 14256 | 025653 | LO4 | LDB | TAP4 |
| 0745 | 14257 | 015423 | CPB | PBPTR | POINT TO START OF LOADED AREA |
| 0746 | 14260 | 066266 | JMP | LO5 | END OF LOADED AREA? |
| 0747 | 14261 | 074517 | LDA | B,I | YES |
| 0748 | 14262 | 001663 | ADA | OFFSET | GET LOADED LINE NUMBER |
| 9749 | 14263 | 074557 | STA | B,I | ADD OFFSET |
| 0750 | 14264 | 061007 | JSM | NEXTA | STORE NEW LINE # |
| 0751 | 14265 | 066257 | JMP | LO4+1 | COMPUTE ADDR OF NEXT LINE |
| 0752 | 14266 | 025647 | LO5 | LDB | TAP1 |
| 0753 | 14267 | 035423 | STB | PBPTR | |
| 0754 | 14270 | 170402 | RET | | RESTORE END OF PROGR POINTER |
| 0755* | | | | | |
| 0756* | | | | | |
| 0757* | | | | | |
| 0758 | 14271 | 035660 | LO6 | STB | TAP7 |
| 0759 | 14272 | 025671 | LDB | IDBO+5 | START ADDR |
| 0760 | 14273 | 005663 | ADB | OFFSET | END LINE # |
| 0761 | 14274 | 004161 | ADB | MAXSN | |
| 0762 | 14275 | 074112 | SBM | *+2 | FINAL LINE # |
| 0763 | 14276 | 164273 | JMP | E5-1,I | EXCEED 9999? |
| 0764 | 14277 | 066177 | JMP | COMON | YES |
| 0765* | | | | | |
| 0766* | | | | | |
| 0767* | | | | | |
| 0768* | | | | | |
| 0769* | | | | | |
| 0770 | 14300 | 062457 | SFLOC | JSM | UNITC |
| 0771 | 14301 | 021472 | LDA | TEMPS | |
| 0772 | 14302 | 011421 | CPA | PRADD | END OF STATEMENT? |
| 0773 | 14303 | 066312 | JMP | SF2 | YES, PROCESS |
| 0774 | 14304 | 062510 | JSM | FILEC | NO |
| 0775 | 14305 | 031644 | STA | TFILE | |
| 0776 | 14306 | 074742 | SBR | 16 | |
| 0777 | 14307 | 035660 | STB | TAP7 | |
| 0778 | 14310 | 035661 | STB | TAP8 | |
| 0779 | 14311 | 067123 | JMP | LOCF2 | LOC FILE AND RET |
| 0780 | 14312 | 074742 | SF2 | SBR | 16 |
| 0781 | 14313 | 035660 | STB | TAP7 | |
| 0782 | 14314 | 635661 | STB | TAP8 | |
| 0783 | 14315 | 035644 | STB | TFILE | FILE 0 |
| 0784 | 14316 | 067123 | JMP | LOCF2 | |
| 0785* | | | | | |
| 0786* | | | | | |
| 0787* | | | | | |
| 0788* | | | | | |

PAGE 254

9830A TAPE CASSETTE SYSTEM

0789*
 0790* ALSTOR — ALL STORE EXECUTION ENTRY
 0791*
 0792*
 0793 14317 060771 ASTOR JSM GTOPP GET OPERAND
 0794 14320 072510 RZA STORP SKIP IF OPERAND GIVEN
 0795 14321 045472 ISZ TEMPS
 0796 14322 121472 LDA TEMPS,I
 0797 14323 013732 CPA DATOP DATA OPCODE?
 0798 14324 067216 JMP SDATA YES
 0799 14325 013733 CPA KEYOP KEY OPCODE?
 0800 14326 067403 JMP SKEY YES
 0801 14327 013734 CPA BINOP BINARY OPCODE?
 0802 14330 167755 JMP ERR0,I
 0803 14331 055472 DSZ TEMPS
 0804*
 0805*
 0806* STORP — STORE PROGRAM
 0807*
 0808*
 0809 14332 062300 STORP JSM SFLOC
 0810 14333 062071 JSM LO3-1
 0811 14334 063025 JSM WPERM
 0812 14335 025421 LDB PRADD
 0813 14336 163756 JSM DCM1A,I
 0814 14337 121424 LDA PBUFF,I
 0815 14340 070072 SAM *+1,C DECOMPILE
 0816 14341 025660 LDB TAP7 ASSUME FIRST LINE #
 0817 14342 074110 SZB *+2 FIRST LINE #
 0818 14343 021660 LDA TAP7
 0819 14344 025661 LDB TAP8
 0820 14345 076110 RZB *+2 SECOND LINE #
 0821 14346 024056 LDB 1E4 SKIP IF GIVEN
 0822 14347 060730 JSM GETAD ASSUME LAST LINE IN PROGRAM
 0823 14350 015477 CPB TEMP4 GET ADDR LIMITS
 0824 14351 067242 JMP ERR5
 0825 14352 121477 LDA TEMP4,I
 0826 14353 070072 SAM *+1,C NULL PROGRAM
 0827 14354 031670 STA IDB0+4
 0828 14355 021477 LDA TEMP4 IN FILE ID
 0829 14356 031472 STA TEMPS START OF STORE AREA
 0830 14357 031650 STA TAP2 LOW LIMIT
 0831 14360 035660 STB TAP7 LOW LIMIT
 0832 14361 060602 JSM NEXT1-4 HIGH LIMIT
 0833 14362 001466 ADA LOCL1
 0834 14363 013746 CPA TCOM
 0835 14364 066366 JMP *+2 IS IT COMMON?
 0836 14365 066373 JMP *+6 YES
 0837 14366 025405 LDB FWUP NO
 0838 14367 074076 TCB START OF USER AREA
 0839 14370 005424 ADB PBUFF
 0840 14371 035672 STB IDB0+6 FIND SIZE OF COMMON
 0841 14372 066375 JMP *+3 IN FILE ID BUFF
 0842 14373 074742 SBR 16
 0843 14374 035672 STB IDB0+6 ZERO COMMON
 0844 14375 025720 LDB LASTN LAST LINE #

37

| | | | | | |
|-------|-----------------|--------|-------|---------|---------------------------|
| 0845 | 14376 | 035671 | STB | IDB0+5 | IN FILE ID BUFF |
| 0846 | 14377 | 025660 | LDB | TAP7 | TEMPR, END ADDR |
| 0847 | 14400 | 021650 | LDA | TAP2 | TEMPR, START ADDR |
| 0848 | 14401 | 063237 | JSM | SDAT2 | |
| 0849 | 14402 | 063463 | JSM | SECCK | |
| 0850 | 14403 | 024020 | LDB | .3 | UNSEC SOURCE |
| 0851 | 14404 | 070113 | SAP | *+2 | |
| 0852 | 14405 | 024036 | LDB | .23 | SEC SOURCE |
| 0853 | 14406 | 035666 | STB | IDB0+2 | |
| 0854 | 14407 | 067231 | JMP | SDAT3+2 | |
| 0855* | | | | | |
| 0856* | | | | | |
| 0857* | MERGE EXECUTION | | | | |
| 0858* | | | | | |
| 0859* | | | | | |
| 0860 | 14410 | 025472 | MERGE | LDB | TEMPS |
| 0861 | 14411 | 004133 | ADB | M2 | ADDR OF MERGE LINE |
| 0862 | 14412 | 074517 | LDA | B,I | MERGE LINE NO. |
| 0863 | 14413 | 070072 | SAM | *+1,C | CLEAR SEC BIT |
| 0864 | 14414 | 031630 | STA | TREGE | TEMPOR |
| 0865 | 14415 | 062070 | JSM | LOAD2 | |
| 0866 | 14416 | 021661 | LDA | TAP8 | |
| 0867 | 14417 | 072310 | RZA | LO7-2 | LNx2 GIVEN |
| 0868 | 14420 | 021416 | LDA | MODE | |
| 0869 | 14421 | 072310 | RZA | LO7 | DO NOT EXECUTE |
| 0870 | 14422 | 021630 | LDA | TREGE | |
| 0871 | 14423 | 000012 | ADA | .1 | POINT TO NEXT LINE |
| 0872 | 14424 | 031661 | STA | TAP8 | LINE FOR START EXECUTION |
| 0873 | 14425 | 070742 | SAR | 16 | |
| 0874 | 14426 | 031416 | STA | MODE | EXECUTE |
| 0875 | 14427 | 062130 | JSM | LOAD7 | |
| 0876 | 14430 | 060645 | JSM | MINIT | |
| 0877 | 14431 | 021670 | LDA | IDB0+4 | START L.N. |
| 0878 | 14432 | 001663 | ADA | OFFSET | |
| 0879 | 14433 | 061075 | JSM | FNDPS | FIND START ADDR |
| 0880 | 14434 | 066442 | JMP | MERG1 | |
| 0881 | 14435 | 066437 | JMP | *+2 | |
| 0882 | 14436 | 066147 | JMP | ERR8 | OVERLAY ERROR |
| 0883 | 14437 | 035650 | STB | TAP2 | SAVE ADDR |
| 0884 | 14440 | 062167 | JSM | LOAD4 | |
| 0885 | 14441 | 066062 | JMP | LOAD9 | |
| 0886 | 14442 | 062271 | MERG1 | JSM | LO6 |
| 0887 | 14443 | 066062 | JMP | LOAD9 | |
| 0888* | | | | | |
| 0889* | | | | | |
| 0890* | LINK EXECUTION | | | | |
| 0891* | | | | | |
| 0892* | | | | | |
| 0893 | 14444 | 062070 | LINK | JSM | LOAD2 |
| 0894 | 14445 | 062102 | | JSM | LOAD8 |
| 0895 | 14446 | 021660 | | LDA | TAP7 |
| 0896 | 14447 | 062160 | | JSM | LOAD3 |
| 0897 | 14450 | 021416 | | LDA | MODE |
| 0898 | 14451 | 070150 | | SZA | *+3 |
| 0899 | 14452 | 060651 | | JSM | INITS |
| 0900 | 14453 | 164174 | | JMP | RDYDA,I |
| | | | | | RESET STACKS |
| | | | | | KEYB-RET TO SYSTEM- DONOT |
| | | | | | EXECUTE |

PAGE 256

9830A TAPE CASSETTE SYSTEM

| | | | | | |
|-------|------------------------|--------|-------|--------|------------|
| 0901 | 14454 | 021661 | LDA | TAPR | SEC LINE # |
| 0902 | 14455 | 031725 | STA | NXTST | |
| 0903 | 14456 | 164005 | JMP | XECA,I | |
| 0904* | | | | | |
| 0905* | | | | | |
| 0906* | UNIT NO CONFIGURATION | | | | |
| 0907* | | | | | |
| 0908* | | | | | |
| 0909 | 14457 | 060771 | UNITC | JSM | GTOPP |
| 0910 | 14460 | 072510 | | RZA | F5 |
| 0911 | 14461 | 045472 | | ISZ | TEMPS |
| 0912 | 14462 | 021472 | | LDA | TEMPS |
| 0913 | 14463 | 011421 | | CPA | PRADD |
| 0914 | 14464 | 066472 | | JMP | F5 |
| 0915 | 14465 | 121472 | | LDA | TEMPS,I |
| 0916 | 14466 | 050065 | | AND | OPMSK |
| 0917 | 14467 | 013742 | | CPA | NUMS |
| 0918 | 14470 | 066475 | | JMP | GUNT |
| 0919 | 14471 | 055472 | | DSZ | TEMPS |
| 0920 | 14472 | 023735 | F5 | LDA | B120K |
| 0921 | 14473 | 031645 | | STA | TUNIT |
| 0922 | 14474 | 170402 | | RET | |
| 0923 | 14475 | 061217 | GUNT | JSM | FETCH |
| 0924 | 14476 | 160225 | | JSM | FLTRA,I |
| 0925 | 14477 | 164342 | UOVF | JMP | E40+3,I |
| 0926 | 14500 | 075752 | | SBM | *-1 |
| 0927 | 14501 | 074117 | | LDA | B |
| 0928 | 14502 | 004142 | | ADB | M11 |
| 0929 | 14503 | 075613 | | SBP | UOVF |
| 0930 | 14504 | 070204 | | SAL | 12 |
| 0931 | 14505 | 072110 | | RZA | *+2 |
| 0932 | 14506 | 066477 | | JMP | UOVF |
| 0933 | 14507 | 066473 | | JMP | F5+1 |
| 0934* | | | | | |
| 0935* | | | | | |
| 0936* | FILE NO. CONFIGURATION | | | | |
| 0937* | | | | | |
| 0938* | | | | | |
| 0939 | 14510 | 061217 | FILEC | JSM | FETCH |
| 0940 | 14511 | 160225 | | JSM | FLTRA,I |
| 0941 | 14512 | 066514 | | JMP | *+2 |
| 0942 | 14513 | 074153 | | SBP | *+3 |
| 0943 | 14514 | 160205 | ERR1 | JSM | AERRA,I |
| 0944 | 14515 | 177710 | | DEC | -56 |
| 0945 | 14516 | 074117 | | LDA | B |
| 0946 | 14517 | 170402 | | RET | |
| 0947* | | | | | |
| 0948* | | | | | |
| 0949* | FINDE — FIND EXECUTION | | | | |
| 0950* | | | | | |
| 0951* | | | | | |
| 0952 | 14520 | 062457 | FINDE | JSM | UNITC |
| 0953 | 14521 | 070542 | | SAR | 12 |
| 0954 | 14522 | 062554 | | JSM | FITAS |
| 0955 | 14523 | 000012 | | ADA | .1 |
| 0956 | 14524 | 027744 | | LDB | CASIN |

GET OPERAND
NO # PRESENT

IS IT #?
GET UNIT

ASSUME INTERNAL
UNIT TEMPOR.

EVALUATE FORMULA
CONVERT TO INTEGER

SAVE UNIT IN A REG
UNIT<11?

POSITION UNIT NO.

EVALUATE FORMULA
CONVERT TO INTEGER

RET ON POSITIVE
WRONG FILE NO.

WITH FILE NO IN A REG

GET UNIT NO.
POSITION UNIT

FIG. TABLE ADDR OF SERV.SUBR.
CASST INTRPT SUBRT ADDR

PAGE 257 9830A TAPE CASSETTE SYSTEM

| | | | | | | |
|-------|-------|--------|-------|-----|---------|--|
| 0957 | 14525 | 070577 | | STB | A,I | |
| 0958 | 14526 | 062510 | | JSM | FILEC | GET FILE |
| 0959 | 14527 | 031644 | | STA | TFILE | |
| 0960 | 14530 | 063106 | FIND3 | JSM | RIFS | |
| 0961* | | | | | | |
| 0962* | | | | | | |
| 0963* | | | | | | |
| 0964 | 14531 | 025664 | DIREC | LDB | FILEN | FILE NO.; FROM ID BUFF |
| 0965 | 14532 | 074076 | | TCB | | MAKE FILE NO READ NEG. |
| 0966 | 14533 | 005644 | | ADB | TFILE | FILE WANTED |
| 0967 | 14534 | 004132 | | ADB | M1 | SUBTR, ONE TO STOP FRONT OF
CERRECT |
| 0968 | 14535 | 062552 | | JSM | FITA3 | |
| 0969 | 14536 | 070577 | | STB | A,I | STORE FILE DIFF IN STATUS |
| 0970 | 14537 | 076210 | | RZB | DI3 | |
| 0971 | 14540 | 063031 | | JSM | HALT | FILE LOCATED |
| 0972 | 14541 | 173741 | DI2 | CLF | 1 | TURN ON INTERPT |
| 0973 | 14542 | 164172 | | JMP | XEC4A,I | BACK TO SYSTEM |
| 0974 | 14543 | 074212 | DI3 | SBM | *+4 | |
| 0975 | 14544 | 023747 | | LDA | RCFF | READ CONTRL FORW FASR-T |
| 0976 | 14545 | 063033 | | JSM | COUTP+1 | |
| 0977 | 14546 | 066541 | | JMP | DI2 | |
| 0978 | 14547 | 023750 | | LDA | RCRF | READ CONTRL REV FAST |
| 0979 | 14550 | 063033 | | JSM | COUTP+1 | |
| 0980 | 14551 | 066541 | | JMP | DI2 | |
| 0981* | | | | | | |
| 0982* | | | | | | |
| 0983* | | | | | | |
| 0984 | 14552 | 021645 | FITA3 | LDA | TUNIT | |
| 0985 | 14553 | 070542 | | SAR | 12 | |
| 0986 | 14554 | 000132 | FITAS | ADA | M1 | |
| 0987 | 14555 | 070744 | | SAL | 1 | |
| 0988 | 14556 | 003743 | | ADA | ITBAS | INTRPT TABLE BASE |
| 0989 | 14557 | 170402 | | RET | | |
| 0990* | | | | | | |
| 0991* | | | | | | |
| 0992* | | | | | | |
| 0993* | | | | | | |
| 0994* | | | | | | |
| 0995 | 14560 | 031444 | ISERV | STA | SAVEA | |
| 0996 | 14561 | 035445 | | STB | SAVEB | |
| 0997 | 14562 | 172700 | | SFC | 0 | |
| 0998 | 14563 | 066611 | | JMP | IS2 | MUST BE STOP |
| 0999 | 14564 | 172241 | | LIA | 1 | INPUT INTERPT ADDRESSES |
| 1000 | 14565 | 070504 | | SAL | 6 | |
| 1001 | 14566 | 070242 | | SAR | 6 | CLEAR 6 MSB |
| 1002 | 14567 | 072410 | | RZA | ISE4 | |
| 1003 | 14570 | 023736 | | LDA | KEYBD | |
| 1004 | 14571 | 172160 | | OTA | 16 | |
| 1005 | 14572 | 172741 | | STF | 1 | |
| 1006 | 14573 | 172240 | | LIA | 0 | |
| 1007 | 14574 | 070342 | | SAR | 8 | |
| 1008 | 14575 | 031443 | | STA | KEYFG | |
| 1009 | 14576 | 066651 | | JMP | RESTR | |
| 1010 | 14577 | 074742 | ISE4 | SBR | 16 | |
| 1011 | 14600 | 035446 | | STB | IADDR | TEMPT INTRPT ADDR |
| 1012 | 14601 | 070006 | | RAR | 1 | POSITION NEXT H.P. BIT |

PAGE 258

9830A TAPE CASSETTE SYSTEM

| Address | Hex | Hex | Label | Instruction | Comment |
|---------|-------|--------|-------|-------------|--------------------------------|
| 1013 | 14602 | 045446 | ISZ | IADDR | INCRM COUNT |
| 1014 | 14603 | 071713 | SAP | *-2 | CONTINUE |
| 1015 | 14604 | 021446 | LDA | IADDR | |
| 1016 | 14605 | 062554 | JSM | FITAS | |
| 1017 | 14606 | 000012 | ADA | .1 | FIG. TABLE ADDR OF SERV SUBR |
| 1018 | 14607 | 070537 | LDB | A,I | |
| 1019 | 14610 | 074737 | JMP | B,I | GO TO SERVICE SUBR |
| 1020 | 14611 | 020040 | LDA | .31 | |
| 1021 | 14612 | 031442 | STA | STPFL | |
| 1022 | 14613 | 020104 | LDA | B177 | |
| 1023 | 14614 | 031443 | STA | KEYFG | |
| 1024 | 14615 | 021444 | LDA | SAVEA | |
| 1025 | 14616 | 025445 | LDB | SAVEB | |
| 1026 | 14617 | 170402 | RET | | |
| 1027* | | | | | |
| 1028* | | | | | |
| 1029* | | | | | |
| 1030* | | | | | |
| 1031* | | | | | |
| 1032* | | | | | |
| 1033* | | | | | |
| 1034* | | | | | |
| 1035 | 14620 | 021446 | LDA | IADDR | |
| 1036 | 14621 | 070204 | SAL | 12 | FIGURE SELECT CODE |
| 1037 | 14622 | 063003 | JSM | STAT+2 | GET STATUS |
| 1038 | 14623 | 063007 | JSM | STACK | STATUS CHECK |
| 1039 | 14624 | 021446 | LDA | IADDR | |
| 1040 | 14625 | 062554 | JSM | FITAS | |
| 1041 | 14626 | 070537 | LDB | A,I | LOAD FILE STATUS |
| 1042 | 14627 | 074152 | SBM | *+3 | |
| 1043 | 14630 | 004132 | ADB | M1 | ADD -1 FOR FORW DIRECTIN |
| 1044 | 14631 | 066633 | JMP | IST1 | |
| 1045 | 14632 | 004012 | ADB | .1 | ADD +1 FOR REV DIRECTION |
| 1046 | 14633 | 070577 | STB | A,I | RESTORE UPDATED STATUS |
| 1047 | 14634 | 076110 | RZB | *+2 | |
| 1048 | 14635 | 067020 | JMP | INSTP | FILE LOCATED |
| 1049 | 14636 | 021446 | LDA | IADDR | |
| 1050 | 14637 | 070204 | SAL | 12 | FIG. SELECT, CODE |
| 1051 | 14640 | 074252 | SBM | *+5 | |
| 1052 | 14641 | 043747 | IOR | RCFF | READ-CONTROL-FORW-FAST |
| 1053 | 14642 | 063034 | JSM | COUTP+2 | |
| 1054 | 14643 | 172741 | STF | 1 | COMMAND OUTPUT, INTRPT TURN ON |
| 1055 | 14644 | 066650 | JMP | COINT | |
| 1056 | 14645 | 043750 | IOR | RCRF | READ-CONTROL-REV-FAST |
| 1057 | 14646 | 063034 | JSM | COUTP+2 | |
| 1058 | 14647 | 172741 | STF | 1 | CLEAR CONTROL |
| 1059* | | | | | |
| 1060* | | | | | |
| 1061* | | | | | |
| 1062* | | | | | |
| 1063* | | | | | |
| 1064 | 14650 | 063034 | COINT | JSM | COUTP+2 |
| 1065* | | | | | |
| 1066* | | | | | |
| 1067* | | | | | |
| 1068* | | | | | |

PAGE 259

9830A TAPE CASSETTE SYSTEM

1069*
 1070 14651 062615 RESTR JSM IS3
 1071 14652 173741 CLF 1
 1072 14653 170402 RET
 1073*
 1074*
 1075* BASIC DATA WRITER
 1076*
 1077*
 1078 14654 031653 TWR STA TAP4 STORE LOW ADDRESS LIMIT
 1079 14655 035654 STB TAP5 STORE THE HIGH LIMIT
 1080 14656 021645 TWR1 LDA TUNIT UNIT NO.
 1081 14657 040116 IOR WDFS WRITE DATA FORW SLOW
 1082 14660 070075 SEC *+1,C CLEAR E
 1083 14661 074742 TWR4 SBR 16
 1084 14662 035655 STB TAP6 CLEAR CHECK-SUM REG
 1085 14663 025653 LDB TAP4 LOAD INITIAL POINTER VALUE
 1086 14664 172501 SFS 1
 1087 14665 066664 JMP *-1
 1088 14666 172141 OTA 1
 1089 14667 121653 LDA TAP4,I
 1090 14670 066674 JMP NWL2
 1091 14671 074517 NWL LDA B,I LOAD A TRANSFER WORD
 1092 14672 172501 NWL1 SFS 1 SKIP IF READY
 1093 14673 066672 JMP *-1
 1094 14674 172140 NWL2 OTA 0 WRITE 8 BITS
 1095 14675 073655 SEC NWL1,S WRITE 8 MORE BITS IF E=0
 1096 14676 001655 ADA TAP6 UPDATE CHECK-SUM
 1097 14677 031655 STA TAP6
 1098 14700 070075 SEC *+1,C CLEAR E
 1099 14701 015654 CPB TAP5 AREA TRANSFERRED?
 1100 14702 066705 JMP COUT YES, OUTPUT CHECK-SUM
 1101 14703 077330 RIB NWL WRITE MORE
 1102 14704 066671 JMP NWL JUMP IF B=0
 1103 14705 021655 COUT LDA TAP6 LOAD CHECK-SUM
 1104 14706 172501 SFS 1
 1105 14707 066706 JMP *-1
 1106 14710 172140 OTA 0 OUTPUT 8 BITS
 1107 14711 073655 SEC *-3,S
 1108 14712 025442 LDB STOP
 1109 14713 074150 SZB *+3
 1110 14714 063031 JSM HALT
 1111 14715 067016 JMP ERR3 IMPROPER STOP
 1112 14716 170402 RET
 1113*
 1114*
 1115* BASIC DATA READER
 1116*
 1117*
 1118 14717 031653 TRD STA TAP4 STORE START OF BUFFER AREA
 1119 14720 035654 STB TAP5 STORE THE HIGH LIMIT
 1120 14721 021645 TRD1 LDA TUNIT UNIT NO.
 1121 14722 172141 OTA 1 READ DATA FORW SLOW
 1122 14723 172601 STC 1 ENABLE CASST
 1123 14724 070742 SAR 16
 1124 14725 031655 STA TAP6 CLEAR CHECK SUM REG

PAGE 260

9830A TAPE CASSETTE SYSTEM

| | | | | | | |
|-------|-------|--------|-------|-----|-------|---|
| 1125 | 14726 | 021653 | | LDA | TAP4 | LOAD INITIAL POINTER VALUE |
| 1126 | 14727 | 070075 | TRL | SEC | *+1,C | CLEAR ERROR FLAG |
| 1127 | 14730 | 172501 | | SFS | 1 | SKIP IF DONE |
| 1128 | 14731 | 066730 | | JMP | *-1 | |
| 1129 | 14732 | 176240 | | LIB | 0 | INPUT 8 BITS |
| 1130 | 14733 | 073655 | | SEC | *-3,S | GET NEXT 8 BBITS |
| 1131 | 14734 | 011654 | | CPA | TAP5 | AREA TRANSFERED? |
| 1132 | 14735 | 066742 | | JMP | CHCK | YES, RETURN AFTER CHECKING |
| 1133 | 14736 | 070577 | | STB | A,I | STORE THE READ WORD RESULT |
| 1134 | 14737 | 005655 | | ADB | TAP6 | UPDATE CHECK SUM WORD |
| 1135 | 14740 | 035655 | | STB | TAP6 | |
| 1136 | 14741 | 073330 | | RIA | TRL | READ MORE |
| 1137 | 14742 | 015655 | CHCK | CPB | TAP6 | CHECK-SUM OK? |
| 1138 | 14743 | 070075 | | SEC | *+1,C | CLEAR ERROR FLAG |
| 1139 | 14744 | 172741 | | STF | 1 | TURN OFF CEO |
| 1140 | 14745 | 170402 | | RET | | |
| 1141* | | | | | | |
| 1142* | | | | | | |
| 1143* | | | | | | |
| 1144* | | | | | | |
| 1145* | | | | | | |
| 1146* | | | | | | |
| 1147 | 14746 | 063610 | WRFIL | JSM | RRBOF | |
| 1148 | 14747 | 062776 | | JSM | CSTOP | |
| 1149 | 14750 | 020121 | WRF3 | LDA | RCFS | READ CONTROL FORW SLOW |
| 1150 | 14751 | 063037 | | JSM | BOFCK | |
| 1151 | 14752 | 017740 | | CPB | MARK | |
| 1152 | 14753 | 066755 | | JMP | *+2 | |
| 1153 | 14754 | 066750 | | JMP | WRF3 | KEEP LOOKING |
| 1154 | 14755 | 062656 | | JSM | TWR1 | WRITE ID |
| 1155 | 14756 | 021650 | | LDA | TAP2 | START ADDR |
| 1156 | 14757 | 025660 | | LDB | TAP7 | END ADDR |
| 1157 | 14760 | 004132 | | ADB | M1 | ADJUST COUNTER FOR WRITE ROUTINE |
| 1158 | 14761 | 062654 | | JSM | TWR | WRITE FILE BODY |
| 1159 | 14762 | 172501 | | SFS | 1 | |
| 1160 | 14763 | 066762 | | JMP | *-1 | WAIT |
| 1161 | 14764 | 067031 | | JMP | HALT | |
| 1162* | | | | | | |
| 1163* | | | | | | |
| 1164* | | | | | | |
| 1165* | | | | | | |
| 1166* | | | | | | |
| 1167 | 14765 | 063001 | STATE | JSM | STAT | |
| 1168 | 14766 | 074406 | | RBR | 9 | POSITION POWER ON BIT |
| 1169 | 14767 | 074153 | | SBP | STAT5 | CAST NOT ON OR NOT PRESENT |
| 1170 | 14770 | 074046 | | RBR | 2 | POSITION CLR+OPEN DOOR BITS |
| 1171 | 14771 | 074111 | | SLB | *+2 | |
| 1172 | 14772 | 067014 | STAT5 | JMP | STAC3 | |
| 1173 | 14773 | 074112 | | SBM | *+2 | |
| 1174 | 14774 | 066776 | | JMP | *+2 | |
| 1175 | 14775 | 064357 | | JMP | RETN2 | CLEAR LEADER |
| 1176 | 14776 | 025442 | CSTOP | LDB | STOP | CHECK STOP |
| 1177 | 14777 | 074350 | | SZB | STAT4 | |
| 1178 | 15000 | 067606 | | JMP | GBACK | STOP CASST+GO BACK TO SYSTM |
| 1179* | | | | | | |
| 1180* | | | | | | |

PAGE 261

9830A TAPE CASSETTE SYSTEM

1181*
 1182 15001 021645 STAT LDA TUNIT
 1183 15002 172741 STF 1
 1184 15003 172141 OTA 1
 1185 15004 172741 STF 1
 1186 15005 176241 LIB 1
 1187 15006 170402 STAT4 RET
 1188*
 1189*
 1190* STACK — STATUS CHECK
 1191*
 1192*
 1193 15007 074506 STACK, RBR 11
 1194 15010 074112 SBM *+2
 1195 15011 075651 SLB STAT4
 1196 15012 063020 JSM INSTP
 1197 15013 067016 JMP ERR3
 1198 15014 063031 STAC3 JSM HALT
 1199 15015 173741 CLF 1
 1200 15016 160205 ERR3 JSM AERRA,I
 1201 15017 177706 DEC -58
 1202*
 1203*
 1204* INSTP — STOP FROM INTRPT ROUTINE
 1205*
 1206*
 1207 15020 021446 INSTP LDA IADDR
 1208 15021 070204 SAL 12
 1209 15022 040117 IOR STOPC
 1210 15023 063034 JSM COUTP+2
 1211 15024 066651 JMP RESTR
 1212*
 1213*
 1214* WPERM — IS WRITE PERMITTED?
 1215*
 1216*
 1217 15025 063001 WPERM JSM STAT
 1218 15026 074504 SBL 6
 1219 15027 074353 SBP COUT1
 1220 15030 067014 JMP STAC3
 1221*
 1222*
 1223* HALT SUBR
 1224*
 1225*
 1226 15031 020117 HALT LDA STOPC
 1227*
 1228*
 1229* COMMAND OUTPUT
 1230*
 1231* ENTRY: COMMAND CONSTANT IN A REG.
 1232*
 1233*
 1234 15032 172741 COUTP STF 1
 1235 15033 041645 IOR TUNIT
 1236 15034 172141 OTA 1

TURN OFF INTERPT
 LOAD I/O REG WITH SELECT CODE
 LOAD I/O REG WITH STATUS
 INPUT STATUS INFO

POSITION CLEAR LEADER + OPEN
 DOOR BITS

CHECK CLEAR LEADER
 CHECK OPEN DOOR

STOP CASST
 TURN ON INTERPT

FIG. SEL CODE

POSITION WRITE PERMIT
 NOT PERMITTED

CONSTANT

TURN OFF INTERPT
 OUTPUT COMMAND

PAGE 262

9830A TAPE CASSETTE SYSTEM

| | | | | | | | |
|-------|-------|--------|----------------------------|-----|---------|--------------------------------|--|
| 1237 | 15035 | 172601 | | STC | 1 | | |
| 1238 | 15036 | 170402 | COUT1 | RET | | ENABLE CAST | |
| 1239* | | | | | | | |
| 1240* | | | | | | | |
| 1241* | | | BOFCK — CHECK FOR BOF | | | | |
| 1242* | | | | | | | |
| 1243* | | | | | | | |
| 1244 | 15037 | 063032 | BOFCK | JSM | COUTP | | |
| 1245 | 15040 | 172501 | | SFS | 1 | | |
| 1246 | 15041 | 067046 | | JMP | BSTP | | |
| 1247 | 15042 | 176241 | | LIB | 1 | INPUT CONTRL CHAR | |
| 1248 | 15043 | 074404 | | SBL | 8 | | |
| 1249 | 15044 | 074342 | | SBR | 8 | CLEAR 8 MSB | |
| 1250 | 15045 | 170402 | | RET | | | |
| 1251 | 15046 | 025442 | BSTP | LDB | STOP | | |
| 1252 | 15047 | 075450 | | SZB | BOFCK+1 | WAIT FOR BOF | |
| 1253 | 15050 | 067606 | | JMP | GBACK | STOP CASST + GO BACK TO SYSTEM | |
| 1254* | | | | | | | |
| 1255* | | | | | | | |
| 1256* | | | RIFID — READ IMMDT FILE ID | | | | |
| 1257* | | | | | | | |
| 1258* | | | | | | | |
| 1259 | 15051 | 063031 | RIFID | JSM | HALT | | |
| 1260 | 15052 | 062765 | | JSM | STATE | | |
| 1261 | 15053 | 067055 | | JMP | *+2 | | |
| 1262 | 15054 | 063656 | | JSM | OFFCL | GET OFF CLR LDR | |
| 1263 | 15055 | 020133 | RIFIS | LDA | M2 | | |
| 1264 | 15056 | 031643 | | STA | TAP0 | | |
| 1265 | 15057 | 023751 | | LDA | IDBA | ID BUFF ADDR | |
| 1266 | 15060 | 031653 | | STA | TAP4 | | |
| 1267 | 15061 | 023753 | | LDA | EOBID | END OF BUFF FOR ID | |
| 1268 | 15062 | 031654 | | STA | TAP5 | | |
| 1269 | 15063 | 062776 | RIF4 | JSM | CSTOP | | |
| 1270 | 15064 | 020121 | | LDA | RCFS | READ CONTRL FOR W SLOW | |
| 1271 | 15065 | 063037 | | JSM | BOFCK | | |
| 1272 | 15066 | 017740 | | CPB | MARK | BOF? | |
| 1273 | 15067 | 067073 | | JMP | RD1 | READ ID | |
| 1274 | 15070 | 062765 | | JSM | STATE | | |
| 1275 | 15071 | 067063 | | JMP | RIF4 | | |
| 1276 | 15072 | 170402 | | RET | | CLEAR LEADER | |
| 1277 | 15073 | 062721 | RD1 | JSM | TRD1 | READ FILE ID | |
| 1278 | 15074 | 070114 | | SES | *+2 | ERROR? | |
| 1279 | 15075 | 064357 | | JMP | RETN2 | NO | |
| 1280 | 15076 | 045643 | | ISZ | TAP0 | | |
| 1281 | 15077 | 067103 | | JMP | RIF6 | | |
| 1282 | 15100 | 063031 | ERR4 | JSM | HALT | CHECK SUM ERROR | |
| 1283 | 15101 | 160205 | | JSM | AERRA,I | | |
| 1284 | 15102 | 177705 | | DEC | -59 | | |
| 1285 | 15103 | 063610 | RIF6 | JSM | RRBOF | | |
| 1286 | 15104 | 067057 | | JMP | RIFIS+2 | TRY AGAIN | |
| 1287* | | | | | | | |
| 1288* | | | | | | | |
| 1289* | | | | | | | |
| 1290 | 15105 | 063610 | RRRID | JSM | RRBOF | REV READ BOF | |
| 1291 | 15106 | 063051 | RIFS | JSM | RIFID | READ IMMDT FILE ID | |
| 1292 | 15107 | 067016 | | JMP | ERR3 | CLR LDR | |

PAGE 263

9830A TAPE CASSETTE SYSTEM

| | | | | | |
|-------|-------|--------|-------|-----|---------|
| 1293 | 15110 | 170402 | | RET | |
| 1294* | | | | | |
| 1295* | | | | | |
| 1296* | | | | | |
| 1297* | | | | | |
| 1298* | | | | | |
| | | | LOC | LOC | LOC |
| 1299 | 15111 | 045472 | LOC | ISZ | TEMPS |
| 1300 | 15112 | 121472 | | LDA | TEMPS,1 |
| 1301 | 15113 | 050065 | | AND | OPMSK |
| 1302 | 15114 | 013742 | | CPA | NUMS |
| 1303 | 15115 | 067120 | | JMP | *+3 |
| 1304 | 15116 | 062472 | | JSM | F5 |
| 1305 | 15117 | 067121 | | JMP | *+2 |
| 1306 | 15120 | 062475 | | JSM | GUNT |
| 1307 | 15121 | 062510 | | JSM | FILEC |
| 1308 | 15122 | 031644 | | STA | TFILE |
| 1309 | 15123 | 063106 | LOC | JSM | RIFS |
| 1310 | 15124 | 020133 | | LDA | M2 |
| 1311 | 15125 | 031647 | | STA | TAP1 |
| 1312 | 15126 | 025664 | LOC | LDB | FILEN |
| 1313 | 15127 | 015644 | | CPB | TFILE |
| 1314 | 15130 | 067031 | LOC | JMP | HALT |
| 1315 | 15131 | 074076 | | TCB | |
| 1316 | 15132 | 005644 | | ADB | TFILE |
| 1317 | 15133 | 004132 | | ADB | M1 |
| 1318 | 15134 | 035653 | | STB | TAP4 |
| 1319 | 15135 | 076110 | | RZB | *+2 |
| 1320 | 15136 | 067162 | | JMP | FVERF |
| 1321 | 15137 | 074153 | | SBP | *+3 |
| 1322 | 15140 | 023750 | | LDA | RCRF |
| 1323 | 15141 | 067145 | | JMP | GDIR-1 |
| 1324 | 15142 | 074076 | | TCB | |
| 1325 | 15143 | 035653 | | STB | TAP4 |
| 1326 | 15144 | 023747 | | LDA | RCFF |
| 1327 | 15145 | 031654 | | STA | TAP5 |
| 1328 | 15146 | 021654 | GDIR | LDA | TAP5 |
| 1329 | 15147 | 062776 | | JSM | CSTOP |
| 1330 | 15150 | 063037 | | JSM | BOFCK |
| 1331 | 15151 | 017740 | | CPB | MARK |
| 1332 | 15152 | 067156 | | JMP | GDIR2 |
| 1333 | 15153 | 062765 | | JSM | STATE |
| 1334 | 15154 | 067146 | | JMP | GDIR |
| 1335 | 15155 | 067016 | | JMP | ERR3 |
| 1336 | 15156 | 045653 | GDIR | ISZ | TAP4 |
| 1337 | 15157 | 067146 | | JMP | GDIR |
| 1338 | 15160 | 020160 | | LDA | DEL2 |
| 1339 | 15161 | 072030 | | RIA | * |
| 1340* | | | | | |
| 1341* | | | | | |
| 1342* | | | | | |
| 1343 | 15162 | 063106 | FVERF | JSM | RIFS |
| 1344 | 15163 | 021644 | | LDA | TFILE |
| 1345 | 15164 | 011664 | | CPA | IDB0 |
| 1346 | 15165 | 067031 | | JMP | HALT |
| 1347 | 15166 | 045647 | | ISZ | TAP1 |
| 1348 | 15167 | 067126 | | JMP | LOC |

IS IT #?

ASSUME INTERNAL

GET UNIT NO.
GET FILEFLAG FOR FILE LOCATION
FILE NO. FROM ID BUFFFILE LOCATED, RET
MAKE FIL NO. READ NEG.
FILE WANTED

SAVE DIFF

FILE LOCATED

READ CONTRL REV FAST

MAKE DIFF NEG.

READ CONTRL FORW FAST

LOK-OK FOR BOF
CLR LDRCONTINUE LOOKING
15.6 MS DELAY

READ IMMDET FILE ID

FILE LOCATED, T-RETURN

TRY AGAIN

PAGE 264

9830A TAPE CASSETTE SYSTEM

| | | | | | |
|-------|---------------------------------------|--------|-------|------|--------------------------------|
| 1349 | 15170 | 063031 | JSM | HALT | |
| 1350 | 15171 | 066514 | JMP | ERR1 | FILE NOT FOUND |
| 1351* | | | | | |
| 1352* | | | | | |
| 1353* | LDATA—LOAD DATA EXECUTION | | | | |
| 1354* | | | | | |
| 1355* | | | | | |
| 1356 | 15172 | 063111 | LDATA | JSM | LOC F |
| 1357 | 15173 | 063256 | | JSM | ARYLS |
| 1358 | 15174 | 025652 | | LDB | TAP3 |
| 1359 | 15175 | 015670 | | CPB | IDB0+4 |
| 1360 | 15176 | 067201 | | JMP | *+3 |
| 1361 | 15177 | 160205 | ERR6 | JSM | AERRA,I |
| 1362 | 15200 | 177703 | | DEC | -61 |
| 1363 | 15201 | 070056 | | CMA | |
| 1364 | 15202 | 001665 | | ADA | IDB0+1 |
| 1365 | 15203 | 070112 | | SAM | *+2 |
| 1366 | 15204 | 067242 | | JMP | ERR5 |
| 1367 | 15205 | 063105 | | JSM | RRRID |
| | | | | | REV READ BOF, READ IMMDET FILE |
| 1368 | 15206 | 021650 | | LDA | TAP2 |
| 1369 | 15207 | 025665 | | LDB | IDB0+1 |
| 1370 | 15210 | 070037 | | ADB | A |
| 1371 | 15211 | 062717 | | JSM | TRD |
| 1372 | 15212 | 063031 | | JSM | HALT |
| 1373 | 15213 | 070115 | | SEC | *+2 |
| 1374 | 15214 | 067100 | | JMP | ERR4 |
| 1375 | 15215 | 067654 | | JMP | SYSRE |
| 1376* | | | | | CHECK SUM ERROR |
| 1377* | | | | | SYSTEM RET |
| 1378* | SDATA—STORE DATA | | | | |
| 1379* | | | | | |
| 1380* | | | | | |
| 1381 | 15216 | 063252 | SDATA | JSM | ISTOR |
| 1382 | 15217 | 063256 | | JSM | ARYLS |
| 1383 | 15220 | 025650 | | LDB | TAP2 |
| 1384 | 15221 | 070037 | | ADB | A |
| 1385 | 15222 | 021652 | | LDA | TAP3 |
| 1386 | 15223 | 031670 | | STA | IDB0+4 |
| 1387 | 15224 | 035660 | | STB | TAP7 |
| 1388 | 15225 | 021650 | | LDA | TAP2 |
| 1389 | 15226 | 063237 | | JSM | SDAT2 |
| 1390* | | | | | |
| 1391* | | | | | |
| 1392* | | | | | |
| 1393 | 15227 | 020017 | SDAT3 | LDA | .2 |
| 1394 | 15230 | 031666 | | STA | IDB0+2 |
| 1395 | 15231 | 023751 | | LDA | IDBA |
| 1396 | 15232 | 031653 | | STA | TAP4 |
| 1397 | 15233 | 023754 | | LDA | EBIDW |
| 1398 | 15234 | 031654 | | STA | TAP5 |
| 1399 | 15235 | 062746 | | JSM | WRFIL |
| 1400 | 15236 | 067654 | | JMP | SYSRE |
| 1401* | | | | | ERITE THE FILE |
| 1402* | ENTER: AREG=START ADDR, BREG=END ADDR | | | | |
| 1403* | | | | | SYSTEM RET |
| 1404 | 15237 | 070076 | SDAT2 | TCA | |

PAGE 265

9830A TAPE CASSETTE SYSTEM

| | | | | | |
|-------|---------------------------------|--------|-------|-------------|----------------------------|
| 1405 | 15240 | 070037 | | ADB A | FIND FILE FIZE NEEDED |
| 1406 | 15241 | 076150 | | RZB *+3 | SKIP IF NON-ZERO |
| 1407 | 15242 | 160205 | ERR5 | JSM AERRA,I | WRONG FILE SIZE |
| 1408 | 15243 | 177704 | | DEC -60 | |
| 1409 | 15244 | 075712 | | SBM *-2 | ERROR IF NEG |
| 1410 | 15245 | 035665 | | STB IDB0+1 | |
| 1411 | 15246 | 074076 | | TCB | |
| 1412 | 15247 | 005667 | | ADB IDB0+3 | ABS FILE SIZE ON TAPE |
| 1413 | 15250 | 075512 | | SBM ERR5 | SIZE ERROR |
| 1414 | 15251 | 170402 | | RET | |
| 1415* | | | | | |
| 1416* | | | | | |
| 1417* | ISTOR — INITLZ FOR STORE | | | | |
| 1418* | | | | | |
| 1419* | | | | | |
| 1420 | 15252 | 063111 | ISTOR | JSM LOCF | |
| 1421 | 15253 | 063025 | | JSM WPERM | |
| 1422 | 15254 | 023752 | | LDA IDB4A | |
| 1423 | 15255 | 067647 | | JMP RE3-1 | CLEAR 4 CENTER WORDS + RET |
| 1424* | | | | | |
| 1425* | | | | | |
| 1426* | ARYLS — DATA ARRAY CALCULATIONS | | | | |
| 1427* | | | | | |
| 1428* | | | | | |
| 1429 | 15256 | 021472 | ARYLS | LDA TEMPS | |
| 1430 | 15257 | 011421 | | CPA PRADD | END OF STMT? |
| 1431 | 15260 | 067302 | | JMP ARCOM | YES, MUST BE COMMON |
| 1432 | 15261 | 060771 | | JSM GTOPP | NO, GET ARRAY NAME |
| 1433 | 15262 | 160177 | | JSM SSYMA,I | SEARCH SYMBOL TABLE |
| 1434 | 15263 | 164340 | | JMP E40+1,I | NOT FOUND |
| 1435 | 15264 | 074517 | | LDA B,I | READ NAME AND TYPE |
| 1436 | 15265 | 031661 | | STA TAP8 | TEMPR |
| 1437 | 15266 | 050040 | | AND B37 | |
| 1438 | 15267 | 070042 | | SAR 2 | |
| 1439 | 15270 | 031652 | | STA TAP3 | TEMPR, PRECISION |
| 1440 | 15271 | 004132 | | ADB M1 | |
| 1441 | 15272 | 074517 | | LDA B,I | GET ARRAY ADDR |
| 1442 | 15273 | 031650 | | STA TAP2 | |
| 1443 | 15274 | 000132 | | ADA M1 | |
| 1444 | 15275 | 070537 | | LDB A,I | READ CURR DIMENSION |
| 1445 | 15276 | 021661 | | LDA TAP8 | |
| 1446 | 15277 | 160240 | | JSM MDIMA,I | COMPUTE THE SIZE |
| 1447 | 15300 | 000133 | | ADA M2 | |
| 1448 | 15301 | 170402 | | RET | |
| 1449 | 15302 | 020020 | ARCOM | LDA .3 | |
| 1450 | 15303 | 031652 | | STA TAP3 | SET COMMON TYPE |
| 1451 | 15304 | 021405 | | LDA FWUP | |
| 1452 | 15305 | 031650 | | STA TAP2 | SET START ADDR |
| 1453 | 15306 | 070076 | | TCA | |
| 1454 | 15307 | 025422 | | LDB UKPTR | |
| 1455 | 15310 | 074150 | | SZB *+3 | KEY? |
| 1456 | 15311 | 001425 | | ADA PBUFO | YES |
| 1457 | 15312 | 170402 | | RET | |
| 1458 | 15313 | 001424 | | ADA PBUFF | COMPUTE SIZE OF COMMON |
| 1459 | 15314 | 170402 | | RET | |
| 1460* | | | | | |



| | | | | | | |
|-------|------------------|--------|-------|-----|---------|--------------------------------|
| 1461* | | | | | | |
| 1462* | LKEY — LOAD KEYS | | | | | |
| 1463* | | | | | | |
| 1464* | | | | | | |
| 1465 | 15315 | 063111 | LKEY | JSM | LOCF | |
| 1466 | 15316 | 025421 | | LDB | PRADD | |
| 1467 | 15317 | 163756 | | JSM | DCM1A,I | DECOMPILE |
| 1468 | 15320 | 021666 | | LDA | IDB0+2 | FILE TYPE |
| 1469 | 15321 | 010021 | | CPA | .4 | KEY? |
| 1470 | 15322 | 067331 | | JMP | *+7 | |
| 1471 | 15323 | 013741 | | CPA | .24 | SEC KEY ? |
| 1472 | 15324 | 067327 | | JMP | *+3 | |
| 1473 | 15325 | 160205 | ERR7 | JSM | AERRA,I | TYPE ERROR |
| 1474 | 15326 | 177702 | | DEC | -62 | |
| 1475 | 15327 | 024106 | | LDB | B777 | |
| 1476 | 15330 | 035435 | | STB | SECFL | SET SEC FLAG |
| 1477 | 15331 | 021416 | | LDA | MODE | |
| 1478 | 15332 | 072350 | | RZA | LKEY4 | CALC MODE? |
| 1479 | 15333 | 021427 | | LDA | LNUM | NO |
| 1480 | 15334 | 070070 | | SIA | *+1 | |
| 1481 | 15335 | 031661 | | STA | TAP8 | SAVE CURRENT LINE #+1 |
| 1482 | 15336 | 021422 | | LDA | UKPTR | |
| 1483 | 15337 | 070110 | | SZA | *+2 | KEY MODE |
| 1484 | 15340 | 066147 | | JMP | ERR8 | YES, OVEPLAY ERROR |
| 1485 | 15341 | 061225 | LKEY4 | JSM | LDEF | INSURE NOT IN KEY MODE |
| 1486 | 15342 | 021405 | | LDA | FWUP | |
| 1487 | 15343 | 070076 | | TCA | | |
| 1488 | 15344 | 000166 | | ADA | FWAM | COMPUTE — CURR KEY SIZE |
| 1489 | 15345 | 001665 | | ADA | IDB0+1 | COMPUTE DIFFERENCE (NEW—OLD) |
| 1490 | 15346 | 072110 | | RZA | *+2 | |
| 1491 | 15347 | 067371 | | JMP | LKEY9 | |
| 1492 | 15350 | 070412 | | SAM | LKEY5 | NEW KEYS LARGER? |
| 1493 | 15351 | 160250 | | JSM | OVCHA,I | YES CHECK FOR OVERFLOW |
| 1494 | 15352 | 021405 | | LDA | FWUP | |
| 1495 | 15353 | 031476 | | STA | TEMP3 | SET PROGRAM RELOCATION POINTER |
| 1496 | 15354 | 035470 | | STB | L2T1 | |
| 1497 | 15355 | 061121 | | JSM | MVTOH | MOVE MAINLINE AND COMMON |
| 1498 | 15356 | 025470 | | LDB | L2T1 | |
| 1499 | 15357 | 067364 | | JMP | LKEY6 | |
| 1500 | 15360 | 001405 | LKEY5 | ADA | FWUP | |
| 1501 | 15361 | 031477 | | STA | TEMP4 | COMPUTE NEW END OF KEYS |
| 1502 | 15362 | 025405 | | LDB | FWUP | |
| 1503 | 15363 | 160251 | | JSM | CLP1A,I | CLOSE UP PROGRAM |
| 1504 | 15364 | 021424 | LKEY6 | LDA | PBUFF | |
| 1505 | 15365 | 074017 | | ADA | B | |
| 1506 | 15366 | 031424 | | STA | PBUFF | UPDATE START OF MAINLINE |
| 1507 | 15367 | 005405 | | ADB | FWUP | |
| 1508 | 15370 | 035405 | | STB | FWUP | UPDATE END OF KEY AREA |
| 1509 | 15371 | 063105 | LKEY9 | JSM | RRRID | REV READ BOF; READ IMMDT FIELD |
| 1510 | 15372 | 020166 | | LDA | FWAM | START OF KEY AREA |
| 1511 | 15373 | 025665 | | LDB | IDB0+1 | FILE SIZE |
| 1512 | 15374 | 070037 | | ADB | A | END OF KEY AREA |
| 1513 | 15375 | 062717 | | JSM | TRD | READ THE FILE |
| 1514 | 15376 | 063031 | | JSM | HALT | |
| 1515 | 15377 | 070115 | | SEC | *+2 | |
| 1516 | 15400 | 067100 | | JMP | ERR4 | CHECK SUM ERROR |

PAGE 267

9830A TYPE CASSETTE SYSTEM

| | | | | | |
|-------|--------------------|--------|------|---------|-----------------------------------|
| 1517 | 15401 | 060651 | JSM | INITS | RESET STACKS |
| 1518 | 15402 | 066062 | JMP | LOAD9 | |
| 1519* | | | | | |
| 1520* | | | | | |
| 1521* | SKEY — STORE KEYS | | | | |
| 1522* | | | | | |
| 1523* | | | | | |
| 1524 | 15403 | 063252 | SKEY | JSM | ISTOR |
| 1525 | 15404 | 025421 | LDB | PRADD | |
| 1526 | 15405 | 163756 | JSM | DCM1A,I | |
| 1527 | 15406 | 063463 | JSM | SECCK | |
| 1528 | 15407 | 024021 | LDB | .4 | UNSEC KEY |
| 1529 | 15410 | 070113 | SAP | *+2 | |
| 1530 | 15411 | 027741 | LDB | .24 | SEC KEY |
| 1531 | 15412 | 035666 | STB | IDB0+2 | INFILE ID BUFF |
| 1532 | 15413 | 020166 | LDA | FWAM | START OF KEY AREA |
| 1533 | 15414 | 031650 | STA | TAP2 | TEMPOR |
| 1534 | 15415 | 025405 | LDB | FWUP | END OF KEY AREA+1 |
| 1535 | 15416 | 035660 | STB | TAP7 | END OF KEY AREA |
| 1536 | 15417 | 063237 | JSM | SDAT2 | |
| 1537 | 15420 | 067231 | JMP | SDAT3+2 | |
| 1538* | | | | | |
| 1539* | | | | | |
| 1540* | LBIN — LOAD BINARY | | | | |
| 1541* | | | | | |
| 1542* | | | | | |
| 1543 | 15421 | 063111 | LBIN | JSM | LOCF |
| 1544 | 15422 | 021403 | LDA | MAW | POINTER |
| 1545 | 15423 | 025665 | LDB | IDB0+1 | FILE SIZE |
| 1546 | 15424 | 074076 | TCB | | |
| 1547 | 15425 | 074017 | ADA | B | FIG. MEMORY START ADDR |
| 1548 | 15426 | 070137 | LDB | A | SAVE A |
| 1549 | 15427 | 070076 | TCA | | |
| 1550 | 15430 | 001426 | ADA | PBPTO | POINTER |
| 1551 | 15431 | 070112 | SAM | *+2 | |
| 1552 | 15432 | 064561 | JMP | MER8 | OVERFLOW |
| 1553 | 15433 | 021666 | LDA | IDB0+2 | FILE TYPE |
| 1554 | 15434 | 010012 | CPA | .1 | BINARY? |
| 1555 | 15435 | 067443 | JMP | LB12 | YES, UNSEC |
| 1556 | 15436 | 012034 | CPA | .21 | SEC BIN? |
| 1557 | 15437 | 067441 | JMP | *+2 | |
| 1558 | 15440 | 067325 | JMP | ERR7 | |
| 1559 | 15441 | 020106 | LDA | B777 | |
| 1560 | 15442 | 031435 | STA | SECFL | SET SEC FLAG |
| 1561 | 15443 | 021427 | LB12 | LDA | PRESENT L.N. |
| 1562 | 15444 | 070070 | SIA | *+1 | |
| 1563 | 15445 | 031661 | STA | TAP8 | |
| 1564 | 15446 | 076070 | RIB | *+1 | |
| 1565 | 15447 | 035404 | STB | LWAM | POINTER |
| 1566 | 15450 | 063105 | JSM | RRRID | REV READ BOF; READ IMMDET FILE ID |
| 1567 | 15451 | 021404 | LDA | LWAM | |
| 1568 | 15452 | 025665 | LDB | IDB0+1 | FILE SIZE |
| 1569 | 15453 | 070037 | ADB | A | END OF BIN |
| 1570 | 15454 | 062717 | JSM | TRD | READ THE FILE |
| 1571 | 15455 | 063031 | JSM | HALT | |
| 1572 | 15456 | 070115 | SEC | *+2 | |

384

372

1,444,141

372

PAGE 268

9830A TAPE CASSETTE SYSTEM

| | | | | | | |
|-------|-------|--------|-------|-----|---------|---------------------------------|
| 1573 | 15457 | 067100 | | JMP | ERR4 | CHECK SUM ERROR |
| 1574 | 15460 | 055404 | | DSZ | LWAM | |
| 1575 | 15461 | 060645 | | JSM | MINIT | SCRATCH SYMBOL TABLE |
| 1576 | 15462 | 066062 | | JMP | LOAD9 | |
| 1577* | | | | | | |
| 1578* | | | | | | |
| 1579* | | | | | | |
| 1580* | | | | | | |
| 1581* | | | | | | |
| 1582 | 15463 | 021435 | SECCK | LDA | SECFL | |
| 1583 | 15464 | 050106 | | AND | B777 | |
| 1584 | 15465 | 070150 | | SZA | *+3 | |
| 1585 | 15466 | 160205 | | JSM | AERRA,I | SEC NOT ALLOWED |
| 1586 | 15467 | 177707 | | DEC | -57 | |
| 1587 | 15470 | 021435 | | LDA | SECFL | RET WITH SEC FLG IN A |
| 1588 | 15471 | 170402 | | RET | | |
| 1589* | | | | | | |
| 1590* | | | | | | |
| 1591* | | | | | | |
| 1592* | | | | | | |
| 1593* | | | | | | |
| 1594 | 15472 | 024013 | WRZE | LDB | B400 | |
| 1595 | 15473 | 074744 | | SBL | 1 | CONVERT WORD COUNT TO CH. COUNT |
| 1596 | 15474 | 074076 | | TCB | | NEG. COUNT |
| 1597 | 15475 | 021645 | | LDA | TUNIT | |
| 1598 | 15476 | 040116 | | IOR | WDFS | WRITE DATA FORW SLOW |
| 1599 | 15477 | 172501 | | SFS | 1 | |
| 1600 | 15500 | 067477 | | JMP | *-1 | |
| 1601 | 15501 | 172141 | | OTA | 1 | |
| 1602 | 15502 | 070742 | WRZE1 | SAR | 16 | |
| 1603 | 15503 | 172501 | | SFS | 1 | |
| 1604 | 15504 | 067503 | | JMP | *-1 | |
| 1605 | 15505 | 172140 | | OTA | 0 | |
| 1606 | 15506 | 077630 | | PIB | WRZE1 | |
| 1607 | 15507 | 170402 | | RET | | |
| 1608* | | | | | | |
| 1609* | | | | | | |
| 1610* | | | | | | |
| 1611* | | | | | | |
| 1612* | | | | | | |
| 1613 | 15510 | 062457 | MARKE | JSM | UNITC | GET UNIT NO. |
| 1614 | 15511 | 063025 | | JSM | WPERM | |
| 1615 | 15512 | 062765 | | JSM | STATE | GET STATUS |
| 1616 | 15513 | 067631 | | JMP | MAR3 | NO CLEAR LEADER |
| 1617 | 15514 | 063524 | | JSM | MAR4 | |
| 1618 | 15515 | 063656 | MAR2 | JSM | OFFCL | GET OFF CLEAR LEADER |
| 1619 | 15516 | 020116 | | LDA | WDFS | WRITE — DATA — FORW — SLOW |
| 1620 | 15517 | 063032 | | JSM | COUTP | COMMAND OUTPUT |
| 1621 | 15520 | 063472 | | JSM | WRZE | WRITE 256 WORDS OF 0 |
| 1622 | 15521 | 172501 | | SFS | 1 | |
| 1623 | 15522 | 067521 | | JMP | *-1 | STILL WRITING |
| 1624 | 15523 | 067547 | | JMP | MCONT | |
| 1625* | | | | | | |
| 1626* | | | | | | |
| 1627* | | | | | | |
| 1628 | 15524 | 062510 | MAR4 | JSM | FILEC | GET NO. OF FILES TO BE MARKED |

PAGE 269

9830A TAPE CASSETTE SYSTEM

| | | | | | |
|-------|-------|--------|-------|-------------|-----------------------------------|
| 1629 | 15525 | 000012 | | ADA .1 | WRITE N#1 FILES |
| 1630 | 15526 | 070076 | | TCA | MAKE FILE COUNT NEG. |
| 1631 | 15527 | 031644 | | STA TFILE | |
| 1632 | 15530 | 062510 | | JSM FILEC | GET LENGTH OF FILE |
| 1633 | 15531 | 031653 | | STA TAP4 | ABS SIZE |
| 1634 | 15532 | 070137 | | LDB A | |
| 1635 | 15533 | 074002 | | SBR 1 | FIND LENGTH/2 |
| 1636 | 15534 | 000135 | | ADA M4 | |
| 1637 | 15535 | 070113 | | SAP *+2 | FILE TOO SMALL |
| 1638 | 15536 | 066514 | | JMP ERR1 | |
| 1639 | 15537 | 021653 | | LDA TAP4 | |
| 1640 | 15540 | 074017 | | ADA B | |
| 1641 | 15541 | 000045 | | ADA .40 | 40 WORDS ADDITIONAL BUFF |
| 1642 | 15542 | 031657 | | STA TLENG | LENGTH TEMPOR |
| 1643 | 15543 | 063644 | | JSM IDCLR | |
| 1644 | 15544 | 021653 | | LDA TAP4 | ABS SIZE |
| 1645 | 15545 | 031667 | | STA IDB0+3 | |
| 1646 | 15546 | 170402 | | RET | |
| 1647* | | | | | |
| 1648* | | | | | |
| 1649* | | | | | |
| 1650 | 15547 | 023730 | MCONT | LDA WCFS | WRITE—CONTROL—FORW—SLOW |
| 1651 | 15550 | 041645 | | IOR TUNIT | |
| 1652 | 15551 | 172141 | | OTA 1 | |
| 1653 | 15552 | 172601 | | STC 1 | ENABLE CASRT |
| 1654 | 15553 | 023751 | MCON2 | LDA IDBA | POINTER TO BEGIN OF ID |
| 1655 | 15554 | 027754 | | LDB EBIDW | POINT TO END OF ID |
| 1656 | 15555 | 062654 | | JSM TWR | WRITE THE FILE ID |
| 1657 | 15556 | 025657 | | LDB TLENG | LOAD ACTUAL FILE LENGTH |
| 1658 | 15557 | 063473 | | JSM WRZE+1 | WRITE THE FILE |
| 1659 | 15560 | 172501 | | SFS 1 | |
| 1660 | 15561 | 067560 | | JMP *-1 | WAIT, IF STILL WRITING |
| 1661 | 15562 | 172741 | | STF 1 | |
| 1662 | 15563 | 172241 | | LIA 1 | INPUT STATUS |
| 1663 | 15564 | 050122 | | AND B6K | ISOLATE OPEN DOOR+CLR LLR |
| 1664 | 15565 | 070610 | | SZA MC2 | SKIP IF OK |
| 1665 | 15566 | 070506 | | RAR 11 | POSITION STATUS BITS |
| 1666 | 15567 | 070151 | | SLA *+3 | |
| 1667 | 15570 | 063031 | ER3 | JSM HALT | |
| 1668 | 15571 | 067016 | | JMP ERR3 | |
| 1669 | 15572 | 020107 | MC3 | LDA RDRS | READ DAR—TA REV SLOW |
| 1670 | 15573 | 063657 | | JSM OFFCL+1 | GET OFF CLEAR LEADER-EOT |
| 1671 | 15574 | 063610 | | JSM RRBOF | |
| 1672 | 15575 | 063472 | | JSM WRZE | ERASE LAST BOF |
| 1673 | 15576 | 172501 | | SFS 1 | |
| 1674 | 15577 | 067576 | | JMP *-1 | WAIT IF STILL WRITING |
| 1675 | 15600 | 063652 | | JSM REWND+1 | GIVE COMMAND AND RETURN TO SYSTEM |
| 1676 | 15601 | 045664 | MC2 | ISZ IDB0 | INCRM FILE NO. |
| 1677 | 15602 | 045644 | | ISZ TFILE | INCRM FILE COUNTER |
| 1678 | 15603 | 067547 | | JMP MCONT | CONTINUE |
| 1679 | 15604 | 024151 | | LDB M32 | |
| 1680 | 15605 | 063611 | | JSM RRBOF+1 | BYPASS 32 CH. + REV. READ BOF |
| 1681 | 15606 | 063031 | GBACK | JSM HALT | |
| 1682 | 15607 | 067654 | | JMP SYSRE | SYSTEM RET |
| 1683* | | | | | |
| 1684* | | | | | |

PAGE 270

9830A TAPE CASSETTE SYSTEM

```

1685*  RRBOF— REVERS READ BOF
1686*
1687*
1688  15610  024146  RRBOF  LDB  M15
1689  15611  020107      LDA  RDRS      READ—DATA—REV—SLOW
1690  15612  063032      JSM  COUTP
1691  15613  172501      SFS  1
1692  15614  067613      JMP  *-1
1693  15615  172240      LIA  0
1694  15616  077670      RIB  *-3
1695  15617  023731      LDA  RCRS      READ—CONTRL—REV—SLOW
1696  15620  063037      JSM  BOFCK
1697  15621  017740      CPB  MARK      BOF?
1698  15622  067626      JMP  DEL65
1699  15623  062765      JSM  STATE      GET STATUS
1700  15624  067610      JMP  RRBOF      KEEP LOOKING
1701  15625  067016      JMP  ERR3      BOT—CLR LDR
1702  15626  023737  DEL65  LDA  DEL1      75 MS DELAY
1703  15627  072030      RIA  *
1704  15630  170402      RET
1705*
1706*
1707*
1708  15631  063106  MAR3  JSM  RIFS
1709  15632  063610      JSM  RRBOF      READ IMMDT FILE ID
1710  15633  025664      LDB  IDB0      REV. READ BOF
1711  15634  035643      STB  TAP0      SAVE LAST FILE NO.
1712  15635  063524      JSM  MAR4      TEMPOR
1713  15636  025643      LDB  TAP0
1714  15637  035664      STB  IDB0
1715  15640  020121      LDA  RCFS
1716  15641  063037      JSM  BOFCK      READ CONTROL FORW SLOWW
1717  15642  017740      CPB  MARK      BOF?
1718  15643  067553      JMP  MCON2      WRITE MORE FILES
1719*
1720*
1721*  IDCLR— CLEAR ID BUFF
1722*
1723*
1724  15644  023751  IDCLR  LDA  IDBA
1725  15645  170000      CLR
1726  15646  000021      ADA  4      CLEAR FILE NO. — ABS SIZE
1727  15647  170000      CLR
1728  15650  170402  RE3    RET      CLEAR STARTING LN + 3 WORDS
1729*
1730*
1731*  REWND— GIVE COMMAND AND RETURN TO SYSTEM
1732*
1733*
1734  15651  062457  REWND  JSM  UNITC      GET UNIT NO.
1735  15652  020110      LDA  RDRF      READ—DATA—REV—FAST
1736  15653  063032      JSM  COUTP
1737  15654  173741  SYSRE  CLF  1      TURN ON INTRPT
1738  15655  164172      JMP  XEC4A,I      RETURN TO SYSTEM
1739*
1740*

```

PAGE 271 9830A TAPE CASSETTE SYSTEM

| | | | | | |
|-------|------------------------------|--------|-------|-------------|---------------------------|
| 1741* | OFFCL — GET OFF CLEAR LEADER | | | | |
| 1742* | | | | | |
| 1743* | | | | | |
| 1744 | 15656 | 020003 | OFFCL | LDA RDFS | READ DATA FORW SLOW |
| 1745 | 15657 | 024124 | | LDB DLCNT | DELAY COUNT |
| 1746 | 15660 | 035652 | | STB TAP3 | TEMPR |
| 1747 | 15661 | 063032 | | JSM COUTP | COMMAND OUTPUT |
| 1748 | 15662 | 063001 | OFI | JSM STAT | GET STATUS |
| 1749 | 15663 | 074544 | | SBL 5 | POSITION CLEAR LEADER BIT |
| 1750 | 15664 | 075213 | | SBP RE3 | |
| 1751 | 15665 | 045652 | | ISZ TAP3 | INCRM DELAY |
| 1752 | 15666 | 067662 | | JMP OFI | WAIT |
| 1753 | 15667 | 067016 | | JMP ERR3 | MUST BE EOT |
| 1754* | | | | | |
| 1755* | | | | | |
| 1756* | LISTE — TAPE LIST EXECUTION | | | | |
| 1757* | | | | | |
| 1758* | | | | | |
| 1759 | 15670 | 062457 | LISTE | JSM UNITC | |
| 1760 | 15671 | 063106 | | JSM RIFS | |
| 1761 | 15672 | 061344 | | JSM CRLF1 | |
| 1762 | 15673 | 063031 | LI2 | JSM HALT | |
| 1763 | 15674 | 025664 | | LDB IDB0 | |
| 1764 | 15675 | 063720 | | JSM PRIN | PRINT FILE NO. |
| 1765 | 15676 | 025666 | | LDB IDB0+2 | FILE TYPE |
| 1766 | 15677 | 063720 | | JSM PRIN | |
| 1767 | 15700 | 025667 | | LDB IDB0+3 | RPINT ABS FILE SIZE |
| 1768 | 15701 | 063720 | | JSM PRIN | CUR SIZE |
| 1769 | 15702 | 025665 | | LDB IDB0+1 | |
| 1770 | 15703 | 063720 | | JSM PRIN | START LINE NO. |
| 1771 | 15704 | 025670 | | LDB IDB0+4 | END LINE NO. |
| 1772 | 15705 | 063720 | | JSM PRIN | |
| 1773 | 15706 | 025671 | | LDB IDB0+5 | NO. OF WORDS IN COMON |
| 1774 | 15707 | 063720 | | JSM PRIN | |
| 1775 | 15710 | 025672 | | LDB IDB0+6 | |
| 1776 | 15711 | 063720 | | JSM PRIN | CL. FLAG FO CRLF |
| 1777 | 15712 | 061344 | | JSM CRLF1 | |
| 1778 | 15713 | 160170 | | JSM CLERA,I | RET TO SYSTEM + REWIND |
| 1779 | 15714 | 031453 | | STA PRFLG | CONTINUE LIST |
| 1780 | 15715 | 063055 | | JSM RIF15 | |
| 1781 | 15716 | 067652 | | JMP REWND+1 | |
| 1782 | 15717 | 067673 | | JMP LI2 | |
| 1783 | 15720 | 035647 | PRIN | STB TAP1 | CLEAR SINGLE LINE BUFF |
| 1784 | 15721 | 160170 | | JSM CLERA,I | |
| 1785 | 15722 | 025647 | | LDB TAP1 | |
| 1786 | 15723 | 045400 | | ISZ BADDR | MAKE ASCII AND STORE |
| 1787 | 15724 | 160210 | | JSM OUTIA,I | CHARACTER COUNT |
| 1788 | 15725 | 020022 | | LDA CCNT | POINT TO FIRST CHAR. |
| 1789 | 15726 | 024212 | | LDB SLBF | PRINT |
| 1790 | 15727 | 065337 | | JMP NDWRT | |
| 1791* | | | | | |
| 1792* | | | | | |
| 1793* | | | | | |
| 1794* | | | | | |
| 1795* | | | | | |
| 1796* | | | | | |

PAGE 272

9830A TAPE CASSETTE SYSTEM

1797*
1798* DEFINITIONS
1799*
1800*
1801 00003 RDFS EQU C6 ZERO
1802 15730 006074 WCFS OCT 6074
1803 15731 005000 RCRS OCT 5000
1804 00110 RDRF EQU B1400
1805 00107 RDRS EQU B1000
1806 01657 TLENG EQU 1657B
1807 00124 DLCNT EQU EXPOP 32000B
1808 15732 077040 DATOP OCT 77040
1809 15733 077000 KEYOP OCT 77000
1810 15734 077200 BINOP OCT 77200
1811 15735 120000 B120K OCT 120000
1812 15736 140000 KEYBD OCT 140000
1813 00004 XECA EQU B10K
1814 00160 B176K EQU M1024
1815 15737 166272 DEL1 DEC -4934
1816 00160 DEL2 EQU M1024
1817 15740 000074 MARK OCT 74
1818 15741 000030 .24 DEC 24
1819 00022 CCNT EQU .5
1820 15742 040000 NUMS OCT 40000
1821 01645 TUNIT EQU 1645B
1822 01644 TFILE EQU 1644B
1823 01446 IADDR EQU ADDR55
1824 15743 040400 ITBAS DEF 40400B
1825 01643 TAP0 EQU 1643B
1826 01647 TAP1 EQU 1647B
1827 01650 TAP2 EQU 1650B
1828 15744 014620 CASIN DEF IST0
1829 01664 FILEN EQU 1664B
1830 01652 TAP3 EQU 1652B
1831 01653 TAP4 EQU 1653B
1832 01654 TAP5 EQU 1654B
1833 01655 TAP6 EQU 1655B
1834 01660 TAP7 EQU 1660B
1835 01661 TAP8 EQU 1661B
1836 15745 001660 TAPAD DEF 1660B
1837 15746 011775 TCOM OCT 11775
1838 01435 SECFL EQU SEC
1839 01663 OFFSET EQU 1663B
1840 15747 004400 RCFF OCT 4400
1841 00121 RCFS EQU B4000
1842 15750 005400 RCRF OCT 5400
1843 15751 001664 IDBA DEF 1664B
1844 01664 IDB0 EQU 1664B
1845 15752 001670 IDB3A DEF 1670B
1846 15753 001700 EOBID DEF 1700B
1847 15754 001677 EBIDW DEF 1677B
1848 00116 WDFS EQU QTOP
1849 00013 RDFF EQU B400
1850 00245 GETSA EQU GTSTA
1851 01442 STOP EQU STPFL
1852 00117 STOPC EQU B2400

END OF ID BUFF FOR WRITE

PAGE 273

9830A TAPE CASSETTE SYSTEM

| | | | | | |
|-------|-------------------|--------|-------|-----|-------------------------|
| 1853 | 15755 | 016747 | ERR0 | DEF | ERROR |
| 1854 | 15756 | 005717 | DCM1A | DEF | 5717B |
| 1855* | | | | | |
| 1856* | TABLE LINKS | | | | |
| 1857* | | | | | |
| 1858 | 15762 | | | ORG | 15762B |
| 1859 | 15762 | 176462 | | ABS | LINK-* |
| 1860 | 15763 | 176334 | | ABS | ASTOR-* |
| 1861 | 15764 | 176534 | | ABS | FINDE-* |
| 1862 | 15765 | 177703 | | ABS | LISTE-* |
| 1863 | 15766 | 177663 | | ABS | REWND-* |
| 1864 | 15767 | 177521 | | ABS | MARKE-* |
| 1865 | 15770 | 176052 | | ABS | ALOAD-* |
| 1866 | 15771 | 176417 | | ABS | MERGE-* |
| 1867 | 15772 | 000000 | | BSS | 1 |
| 1868 | 15773 | 176030 | | ABS | SUBT-* |
| 1869 | 15774 | 162111 | | ABS | B377-* |
| 1870 | 15775 | 176004 | | ABS | CMNDT-* |
| 1871 | 15776 | 000630 | | ABS | SNMT-* |
| 1872 | 15777 | 014000 | | OCT | 14000 |
| 1873* | | | | | |
| 1874* | | | | | |
| 1875* | SYNTAX | | | | |
| 1876* | | | | | |
| 1877 | 16615 | | | ORG | 16615B |
| 1878* | SYNTAX JUMP TABLE | | | | |
| 1879 | 16615 | 000041 | | ABS | MLOST-* |
| 1880 | 16616 | 000064 | | ABS | ELOST-* |
| 1881 | 16617 | 000051 | | ABS | FIMAR-* |
| 1882 | 16620 | 000124 | | ABS | REWL-* |
| 1883 | 16621 | 000123 | | ABS | REWL-* |
| 1884 | 16622 | 000046 | | ABS | FIMAR-* |
| 1885 | 16623 | 000057 | | ABS | ELOST-* |
| 1886 | 16624 | 000032 | | ABS | MLOST-* |
| 1887 | 16625 | 000031 | B31 | OCT | 31 |
| 1888* | SYNTAX TABLE | | | | |
| 1889 | 16626 | 043111 | SNMT | ASC | 2,FIND FIND, OPCOF-DE 7 |
| | 16627 | 047104 | | | |
| 1890 | 16630 | 123522 | | OCT | 123522 OPCODE, R |
| 1891 | 16631 | 042527 | | ASC | 2,EWIN REWIND, OPCODE 5 |
| | 16632 | 044516 | | | |
| 1892 | 16633 | 042245 | | OCT | 42245 D,OPCODE |
| 1893 | 16634 | 046505 | | ASC | 2,MERG MERGE, OPCODE 2 |
| | 16635 | 051107 | | | |
| 1894 | 16636 | 042642 | | OCT | 42642 E,OPCODE |
| 1895 | 16637 | 051524 | | ASC | 2,STOR STORE, OPCODE 10 |
| | 16640 | 047522 | | | |
| 1896 | 16641 | 042650 | | OCT | 42650 E,OPCODE |
| 1897 | 16642 | 046117 | | ASC | 2,LOAD LOAD OPCODE 3 |
| | 16643 | 040504 | | | |
| 1898 | 16644 | 121515 | | OCT | 121515 OPCODE, M |
| 1899 | 16645 | 040522 | | ASC | 1,AR MARK, OPCODE 4 |
| 1900 | 16646 | 045644 | | OCT | 45644 K, OPCODE |
| 1901 | 16647 | 052114 | | ASC | 2,TLIS TLIST, OPCODE 6 |
| | 16650 | 044523 | | | |
| 1902 | 16651 | 052246 | | OCT | 52246 T, OPCODE |

***** CHECK SUM *****

PAGE 274

9830TAPE CASSETTE SYSTEM

| | | | | | | |
|-------|---|--------|-------|-----|----------|------------------------------|
| 1903 | 16652 | 046111 | | ASC | 2, LINK | LINK, OP CODE 11 |
| | 16653 | 047113 | | | | |
| 1904 | 16654 | 164400 | | OCT | 164400 | OP CODE, ET |
| 1905* | | | | | | |
| 1906* | | | | | | |
| 1907* | MERGE, LOAD, STORE SYNTAX CHECK | | | | | |
| 1908* | | | | | | |
| 1909* | | | | | | |
| 1910 | 16655 | 062764 | ML | JSM | UNIT1 | DECRM POINTERS |
| 1911 | 16656 | 062751 | MLOST | JSM | UNIT | |
| 1912 | 16657 | 066661 | | JMP | MA | NO UNIT GIVEN |
| 1913 | 16660 | 062770 | | JSM | COMEL | COMA? NO, EOL |
| 1914 | 16661 | 162773 | MA | JSM | SERVA, I | |
| 1915 | 16662 | 065153 | | JMP | EOLCK | 0- EOL? ACCEPT |
| 1916 | 16663 | 062770 | | JSM | COMEL | FILE NO. GIVEN |
| 1917 | 16664 | 060664 | | JSM | LNUM | GET LINE #1 |
| 1918 | 16665 | 062770 | | JSM | COMEL | |
| 1919 | 16666 | 060664 | | JSM | LNUM | GET LINE #2 |
| 1920 | 16667 | 065151 | | JMP | EOST | |
| 1921* | | | | | | |
| 1922* | | | | | | |
| 1923* | FIND AND MARK SYNTAX CHECK | | | | | |
| 1924* | | | | | | |
| 1925* | | | | | | |
| 1926 | 16670 | 062751 | FIMAR | JSM | UNIT | |
| 1927 | 16671 | 066673 | | JMP | AA | NO UNIT GIVEN |
| 1928 | 16672 | 060510 | | JSM | COMCE | COMA? NO, ERROR |
| 1929 | 16673 | 160202 | AA | JSM | FSCA, I | GET FILE |
| 1930 | 16674 | 025473 | | LDB | TEMP | |
| 1931 | 16675 | 014024 | | CPB | 7 | FIND |
| 1932 | 16676 | 065151 | | JMP | EOST | ACCEPT STATEMENT |
| 1933 | 16677 | 060510 | | JSM | COMCE | |
| 1934 | 16700 | 160202 | | JSM | FSCA, I | |
| 1935 | 16701 | 065151 | | JMP | EOST | |
| 1936* | | | | | | |
| 1937* | | | | | | |
| 1938* | ENTRY FOR ALL LOAD AND STORE | | | | | |
| 1939* | | | | | | |
| 1940* | | | | | | |
| 1941 | 16702 | 021466 | ELOST | LDA | LOCLI | |
| 1942 | 16703 | 050040 | | AND | B37 | |
| 1943 | 16704 | 031473 | | STA | TEMP | TEMPOR, OP CODE |
| 1944 | 16705 | 060456 | | JSM | SBPUD | INCRM POINTER SBPTR |
| 1945 | 16706 | 061055 | | JSM | GNEXT | GET NEXT CHARACTER |
| 1946 | 16707 | 026774 | | LDB | SUBTA | ADDR OF SUB-TABLE |
| 1947 | 16710 | 060546 | | JSM | STEXP | |
| 1948 | 16711 | 066655 | | JMP | ML | LOAD, STORE |
| 1949 | 16712 | 070202 | | SAR | 5 | POSITION KEY OR DATA OP CODE |
| 1950 | 16713 | 050040 | | AND | B37 | |
| 1951 | 16714 | 001473 | | ADA | TEMP | FIGURE EXACT OP CODE |
| 1952 | 16715 | 001473 | | STA | TEMP | |
| 1953* | | | | | | |
| 1954* | | | | | | |
| 1955* | LOADK, STOREK, LOADD, STORED, LOADB, STOREB, SYNTAX CHECK | | | | | |
| 1956* | | | | | | |
| 1957* | | | | | | |

PAGE 275

9830A TAPE CASSETTE SYSTEM

| | | | | | | |
|------|-------|--------|-----|-----|---------|----------------------------|
| 1958 | 16716 | 062754 | | JSM | UNIT2 | |
| 1959 | 16717 | 066741 | | JMP | LA | NO UNIT GIVEN |
| 1960 | 16720 | 060510 | | JSM | COMCE | COMA? NO, ERROR |
| 1961 | 16721 | 162773 | LA1 | JSM | SERVA,I | |
| 1962 | 16722 | 066747 | | JMP | ERROR | EOL AFTER COMA |
| 1963 | 16723 | 025473 | | LDB | TEMP | |
| 1964 | 16724 | 016625 | | CPB | B31 | STORE DATA? |
| 1965 | 16725 | 066731 | | JMP | *+4 | YES |
| 1966 | 16726 | 016775 | | CPB | BB24 | LOAD DATA? |
| 1967 | 16727 | 066731 | | JMP | *+2 | YES |
| 1968 | 16730 | 065151 | | JMP | EOST | DEMAND EOL |
| 1969 | 16731 | 062770 | | JSM | COMEL | COMA? NO, EOL |
| 1970 | 16732 | 061140 | | JSM | LTR | CHECK FOR ARRAY |
| 1971 | 16733 | 066747 | | JMP | ERROR | |
| 1972 | 16734 | 021474 | | LDA | TEMP1 | LETTER; LOAD VARIABLE NAME |
| 1973 | 16735 | 024020 | | LDB | .3 | ASSUME ARRAY |
| 1974 | 16736 | 060451 | | JSM | STROP+1 | PUT IN MEMORY |
| 1975 | 16737 | 021475 | | LDA | TEMP2 | LOAD FOLLOWING CHARACTER |
| 1976 | 16740 | 065151 | | JMP | EOST | |
| 1977 | 16741 | 070742 | LA | SAR | 16 | |
| 1978 | 16742 | 060456 | | JSM | SBPUD | |
| 1979 | 16743 | 066721 | | JMP | LA1 | |

1980*

1981*

REWIND AND TLIST SYNTAX CHECK

1982*

1983*

1984*

| | | | | | | |
|------|-------|--------|-------|-----|---------|------------------|
| 1985 | 16744 | 062751 | REWLI | JSM | UNIT | |
| 1986 | 16745 | 065147 | | JMP | ENDS | |
| 1987 | 16746 | 065151 | | JMP | EOST | ACCEPT STATEMENT |
| 1988 | 16747 | 160205 | ERROR | JSM | AERRA,I | |
| 1989 | 16750 | 177711 | | DEC | -55 | |

1990*

1991*

UNIT SYNTAX CHECK

1992*

1993*

1994*

| | | | | | | |
|------|-------|--------|-------|-----|--------|---------------------|
| 1995 | 16751 | 021466 | UNIT | LDA | LOCL1 | |
| 1996 | 16752 | 050040 | | AND | B37 | RECALL OPCODE |
| 1997 | 16753 | 031473 | | STA | TEMP | OPCODE TEMPORARY |
| 1998 | 16754 | 060456 | UNIT2 | JSM | SBPUD | GET NEW SYNTAX WORD |
| 1999 | 16755 | 061055 | | JSM | GNEXT | GET NEW CHARACTER |
| 2000 | 16756 | 060463 | | JSM | SYMC1 | CHECK FOR # |
| 2001 | 16757 | 012017 | | OCT | 12017 | |
| 2002 | 16760 | 066764 | | JMP | UNIT1 | NO |
| 2003 | 16761 | 066762 | | JMP | FS | YES |
| 2004 | 16762 | 160202 | FS | JSM | FSCA,I | GET UNIT |
| 2005 | 16763 | 064357 | | JMP | RETN2 | |
| 2006 | 16764 | 055706 | UNIT1 | DSZ | SBPTR | |
| 2007 | 16765 | 055400 | | DSZ | BADDR | |
| 2008 | 16766 | 170402 | | RET | | |

2009*

2010*

COMA-EOL CHECK

2011*

2012*

2013*

392

380

1,444,141

380

PAGE 276

9830TAPE CASSETTE SYSTEM

| | | | | | | |
|-------|-------|--------|-------|-----|-----------|-------|
| 2014 | 16767 | 061055 | COMA | JSM | GNEXT | |
| 2015 | 16770 | 060504 | COMEL | JSM | COMCK | COMA? |
| 2016 | 16771 | 065151 | | JMP | EOST | NO |
| 2017 | 16772 | 170402 | | RET | | |
| 2018* | | | | | | |
| 2019* | | | | | | |
| 2020* | | | | | | |
| 2021 | 16773 | 014032 | SERVA | DEF | SERV | |
| 2022 | 16774 | 030046 | SUBTA | DEF | SUBT+SUBT | |
| 2023 | 16775 | 000024 | BB24 | OCT | 24 | |
| 2024 | | | | END | | |

* NO ERRORS*

CROSS-REFERENCE TABLE

[illegible]

PAGE 278

CROSS-REFERENCE TABLE

[illegible]

[illegible]

396

PAGE 280

CROSS-REFERENCE TABLE

| | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|--|
| EBIDW | 1847 | 1397 | 1655 | | | | | | |
| ECALA | 0221 | | | | | | | | |
| ELINA | 0175 | | | | | | | | |
| ELOST | 1941 | 1880 | 1885 | | | | | | |
| ENDS | 0486 | 1986 | | | | | | | |
| ENTEA | 0224 | | | | | | | | |
| ENTFA | 0195 | | | | | | | | |
| EOBID | 1846 | 1267 | | | | | | | |
| EOL | 0044 | 0555 | | | | | | | |
| EOLCK | 0489 | 1915 | | | | | | | |
| EOST | 0487 | 1920 | 1932 | 1935 | 1968 | 1976 | 1987 | 2016 | |
| ER3 | 1667 | | | | | | | | |
| ERR0 | 1853 | 0802 | | | | | | | |
| ERR1 | 0943 | 1350 | 1638 | | | | | | |
| ERR3 | 1200 | 1111 | 1197 | 1292 | 1335 | 1668 | 1701 | 1753 | |
| ERR4 | 1282 | 0724 | 1374 | 1516 | 1573 | | | | |
| ERR5 | 1407 | 0824 | 1366 | 1413 | | | | | |
| ERR6 | 1361 | | | | | | | | |
| ERR7 | 1473 | 0645 | 1558 | | | | | | |
| ERR8 | 0653 | 0681 | 0882 | 1484 | | | | | |
| ERRA | 0183 | | | | | | | | |
| ERROR | 1988 | 1853 | 1962 | 1971 | | | | | |
| ESEC | 0519 | 0516 | | | | | | | |
| ESEC3 | 0522 | 0529 | | | | | | | |
| ESEC4 | 0533 | 0525 | | | | | | | |
| ESEC6 | 0530 | 0523 | | | | | | | |
| EXP | 0407 | 0408 | | | | | | | |
| EXPOP | 0124 | 1807 | | | | | | | |
| F | 0087 | | | | | | | | |
| F5 | 0920 | 0910 | 0914 | 0933 | 1304 | | | | |
| FDAAA | 0228 | | | | | | | | |
| FETCH | 0494 | 0923 | 0939 | | | | | | |
| FILEC | 0939 | 0774 | 0958 | 1307 | 1628 | 1632 | | | |
| FILEN | 1829 | 0964 | 1312 | | | | | | |
| FILLA | 0207 | | | | | | | | |
| FIMAR | 1926 | 1881 | 1884 | | | | | | |
| FIND3 | 0960 | | | | | | | | |
| FINDE | 0952 | 1861 | | | | | | | |
| FIRST | 0373 | | | | | | | | |
| FITA3 | 0984 | 0968 | | | | | | | |
| FITAS | 0986 | 0954 | 1016 | 1040 | | | | | |
| FIX | 0507 | | | | | | | | |
| FLGBT | 0079 | 0531 | | | | | | | |
| FLOAT | 0392 | | | | | | | | |
| FLTFA | 0199 | | | | | | | | |
| FLTRA | 0198 | 0924 | 0940 | | | | | | |
| FLTSA | 0201 | | | | | | | | |
| FN | 0154 | | | | | | | | |
| FND | 0468 | 0678 | | | | | | | |
| FNDIA | 0230 | 0737 | | | | | | | |
| FNDAA | 0229 | 0733 | | | | | | | |
| FNDPS | 0482 | 0665 | 0879 | | | | | | |
| FOPBS | 0212 | | | | | | | | |
| FORMA | 0178 | | | | | | | | |
| FORME | 0342 | | | | | | | | |
| FORMS | 0341 | | | | | | | | |

[illegible]

386

1,444,141

386

PAGE 282

CROSS-REFERENCE TABLE

| | | | | |
|-------|------|------|------|-----------|
| IOSTA | 0231 | | | |
| IS2 | 1020 | 0998 | | |
| IS3 | 1024 | 1070 | | |
| ISE4 | 1010 | 1002 | | |
| ISERV | 0995 | 0184 | | |
| IST0 | 1035 | 1828 | | |
| IST1 | 1046 | 1044 | | |
| ISTOR | 1420 | 1381 | 1524 | |
| ITBAS | 1824 | 0988 | | |
| KCOMN | 0713 | 0691 | 0708 | |
| KEYBD | 1812 | 1003 | | |
| KEYFG | 0330 | 1008 | 1023 | |
| KEYOP | 1809 | 0579 | 0799 | |
| L2T1 | 0353 | 1496 | 1498 | |
| L2T2 | 0354 | | | |
| LA | 1977 | 1959 | | |
| LA1 | 1961 | 1979 | | |
| LASTN | 0404 | 0844 | | |
| LBI2 | 1561 | 1555 | | |
| LBIN | 1543 | 0582 | | |
| LBOP | 0125 | | | |
| LDATA | 1356 | 0578 | | |
| LDEF | 0495 | 1485 | | |
| LETCK | 0470 | | | |
| LT2 | 1762 | 1782 | | |
| LIMIT | 0471 | | | |
| LINK | 0893 | 1859 | | |
| LISTA | 0173 | | | |
| LISTE | 1759 | 1862 | | |
| LKEY | 1465 | 0580 | | |
| LKEY4 | 1485 | 1478 | | |
| LKEY5 | 1500 | 1492 | | |
| LKEY6 | 1504 | 1499 | | |
| LKEY9 | 1509 | 1491 | | |
| LNFD | 0078 | | | |
| LNINC | 0321 | | | |
| LNUM | 0463 | 1917 | 1919 | |
| LO12 | 0655 | 0650 | 0652 | |
| LO3 | 0605 | 0612 | 0810 | |
| LO4 | 0744 | 0734 | 0738 | 0751 |
| LO5 | 0752 | 0746 | | |
| LO6 | 0758 | 0666 | 0886 | |
| LO7 | 0875 | 0867 | 0869 | |
| LOAD2 | 0603 | 0589 | 0865 | 0893 |
| LOAD3 | 0665 | 0593 | 0896 | |
| LOAD4 | 0675 | 0884 | | |
| LOAD7 | 0638 | 0618 | 0621 | 0875 |
| LOAD8 | 0616 | 0590 | 0894 | |
| LOAD9 | 0594 | 0885 | 0887 | 1518 1576 |
| LOADP | 0589 | 0574 | | |
| LOCF | 1299 | 1356 | 1420 | 1465 1543 |
| LOCF1 | 1312 | 1348 | | |
| LOCF2 | 1309 | 0779 | 0784 | |
| LOCF3 | 1314 | | | |
| LOCL1 | 0351 | 0833 | 1941 | 1995 |
| LOCL2 | 0352 | | | |

PAGE 283

CROSS-REFERENCE TABLE

| | | | | | | | |
|-------|------|------|------|------|------|------|-----------|
| LPCK | 0443 | | | | | | |
| LPCKE | 0444 | | | | | | |
| LPOP | 0126 | | | | | | |
| LROMA | 0129 | | | | | | |
| LSTAK | 0305 | | | | | | |
| LSTPT | 0307 | | | | | | |
| LTR | 0485 | 1970 | | | | | |
| LWAM | 0301 | 1565 | 1567 | 1574 | | | |
| M1 | 0130 | 0967 | 0986 | 1043 | 1157 | 1317 | 1440 1443 |
| M10 | 0137 | | | | | | |
| M100 | 0150 | | | | | | |
| M1024 | 0152 | 1814 | 1816 | | | | |
| M11 | 0138 | 0928 | | | | | |
| M12 | 0139 | | | | | | |
| M13 | 0140 | | | | | | |
| M14 | 0141 | | | | | | |
| M15 | 0142 | 1688 | | | | | |
| M2 | 0131 | 0861 | 1263 | 1310 | 1447 | | |
| M23 | 0143 | | | | | | |
| M25 | 0144 | | | | | | |
| M256 | 0151 | | | | | | |
| M3 | 0132 | | | | | | |
| M32 | 0145 | 1679 | | | | | |
| M36 | 0146 | | | | | | |
| M4 | 0133 | 1636 | | | | | |
| M48 | 0147 | | | | | | |
| M5 | 0134 | | | | | | |
| M6 | 0135 | | | | | | |
| M64 | 0148 | | | | | | |
| M80 | 0149 | | | | | | |
| M9 | 0136 | | | | | | |
| MA | 1914 | 1912 | | | | | |
| MAR2 | 1618 | | | | | | |
| MAR3 | 1708 | 1616 | | | | | |
| MAR4 | 1628 | 1617 | 1712 | | | | |
| MARK | 1817 | 1151 | 1272 | 1331 | 1697 | 1717 | |
| MARKE | 1613 | 1864 | | | | | |
| MASK1 | 0159 | | | | | | |
| MASK5 | 0160 | | | | | | |
| MAW | 0300 | 1544 | | | | | |
| MAXIN | 0406 | | | | | | |
| MAXSN | 0153 | 0761 | | | | | |
| MBIN1 | 0412 | | | | | | |
| MBIN2 | 0413 | | | | | | |
| MBOX1 | 0357 | | | | | | |
| MC2 | 1676 | 1664 | | | | | |
| MC3 | 1669 | | | | | | |
| MCON2 | 1654 | 1718 | | | | | |
| MCONT | 1650 | 1624 | 1678 | | | | |
| MDIMA | 0214 | 1446 | | | | | |
| MERB | 0451 | 1552 | | | | | |
| MERG1 | 0886 | 0880 | | | | | |
| MERGE | 0860 | 1866 | | | | | |
| MINIT | 0461 | 0591 | 0876 | 1575 | | | |
| ML | 1910 | 1948 | | | | | |
| MLOST | 1911 | 1879 | 1886 | | | | |

PAGE 284

CROSS-REFERENCE TABLE

[illegible]

PAGE 285

CROSS-REFERENCE TABLE

| | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|--|
| QTOP | 0115 | 0167 | 1848 | | | | | | |
| RBOP | 0123 | | | | | | | | |
| RBUFF | 0366 | | | | | | | | |
| RCFF | 1840 | 0975 | 1052 | 1326 | | | | | |
| RCFS | 1841 | 1149 | 1270 | 1715 | | | | | |
| RCOUT | 0339 | | | | | | | | |
| RCRF | 1842 | 0978 | 1056 | 1322 | | | | | |
| RCRS | 1803 | 1695 | | | | | | | |
| RD1 | 1277 | 1273 | | | | | | | |
| RDFF | 1849 | | | | | | | | |
| RDFS | 1801 | 1744 | | | | | | | |
| RDRF | 1804 | 1735 | | | | | | | |
| RDRS | 1805 | 1669 | 1689 | | | | | | |
| RDYDA | 0172 | 0533 | 0597 | 0900 | | | | | |
| RE3 | 1728 | 1423 | 1750 | | | | | | |
| REED | 0167 | | | | | | | | |
| REEDL | 0345 | | | | | | | | |
| RES | 0423 | | | | | | | | |
| RESB1 | 0459 | | | | | | | | |
| RESBP | 0460 | | | | | | | | |
| RESTR | 1070 | 1009 | 1211 | | | | | | |
| RET2 | 0427 | | | | | | | | |
| RETN2 | 0426 | 0560 | 1175 | 1279 | 2005 | | | | |
| RETN3 | 0425 | | | | | | | | |
| REWLI | 1985 | 1882 | 1883 | | | | | | |
| REWND | 1734 | 1675 | 1781 | 1863 | | | | | |
| RIF4 | 1269 | 1275 | | | | | | | |
| RIF6 | 1285 | 1281 | | | | | | | |
| RIFI5 | 1263 | 1286 | 1780 | | | | | | |
| RIFID | 1259 | 1291 | | | | | | | |
| RIFS | 1291 | 0960 | 1309 | 1343 | 1708 | 1760 | | | |
| RPCK | 0445 | | | | | | | | |
| RPCKE | 0446 | | | | | | | | |
| RPOP | 0121 | | | | | | | | |
| RRBOF | 1688 | 1147 | 1285 | 1290 | 1671 | 1680 | 1700 | 1709 | |
| RRRID | 1290 | 0717 | 1367 | 1509 | 1566 | | | | |
| RSARE | 0298 | | | | | | | | |
| RSLTE | 0379 | | | | | | | | |
| RSPTR | 0424 | | | | | | | | |
| RSTAK | 0422 | | | | | | | | |
| RSTKA | 0164 | | | | | | | | |
| RSULT | 0432 | | | | | | | | |
| RUNA | 0169 | 0599 | | | | | | | |
| S | 0088 | | | | | | | | |
| SALTA | 0213 | | | | | | | | |
| SAVBP | 0458 | | | | | | | | |
| SAVEA | 0331 | 0995 | 1024 | | | | | | |
| SAVEB | 0332 | 0996 | 1025 | | | | | | |
| SBADR | 0400 | | | | | | | | |
| SBPTR | 0393 | 0557 | 2006 | | | | | | |
| SBPUD | 0437 | 0554 | 1944 | 1978 | 1998 | | | | |
| SBUFA | 0205 | | | | | | | | |
| SDAT2 | 1404 | 0848 | 1389 | 1536 | | | | | |
| SDAT3 | 1393 | 0854 | 1537 | | | | | | |
| SDATA | 1381 | 0798 | | | | | | | |
| SEC | 0326 | 1838 | | | | | | | |

| | | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|--|
| SECCK | 1582 | 0849 | 1527 | | | | | | | | | | |
| SECFL | 1838 | 0530 | 0532 | 0647 | 1476 | 1560 | 1582 | 1587 | | | | | |
| SERV | 0553 | 2021 | | | | | | | | | | | |
| SERVA | 2021 | 1914 | 1961 | | | | | | | | | | |
| SF2 | 0780 | 0773 | | | | | | | | | | | |
| SFLAG | 0399 | | | | | | | | | | | | |
| SFLOC | 0770 | 0603 | 0809 | | | | | | | | | | |
| SGNS | 0499 | | | | | | | | | | | | |
| SIGN | 0405 | 0406 | | | | | | | | | | | |
| SKEY | 1524 | 0800 | | | | | | | | | | | |
| SLCOD | 0334 | | | | | | | | | | | | |
| SLEND | 0208 | | | | | | | | | | | | |
| SLWST | 0453 | | | | | | | | | | | | |
| SNFLA | 0203 | | | | | | | | | | | | |
| SNMT | 1889 | 1871 | | | | | | | | | | | |
| SPFLA | 0202 | | | | | | | | | | | | |
| SSBPT | 0401 | | | | | | | | | | | | |
| SSYMA | 0176 | 1433 | | | | | | | | | | | |
| ST | 0383 | | | | | | | | | | | | |
| STAC3 | 1198 | 1172 | 1220 | | | | | | | | | | |
| STACK | 1193 | 1038 | | | | | | | | | | | |
| START | 0168 | 0013 | | | | | | | | | | | |
| STAT | 1182 | 1037 | 1167 | 1217 | 1748 | | | | | | | | |
| STAT4 | 1187 | 1177 | 1195 | | | | | | | | | | |
| STAT5 | 1172 | 1169 | | | | | | | | | | | |
| STATE | 1167 | 1260 | 1274 | 1333 | 1615 | 1699 | | | | | | | |
| STEXP | 0449 | 1947 | | | | | | | | | | | |
| STOP | 1851 | 1108 | 1176 | 1251 | | | | | | | | | |
| STOPC | 1852 | 1209 | 1226 | | | | | | | | | | |
| STORI | 0434 | | | | | | | | | | | | |
| STORE | 0433 | | | | | | | | | | | | |
| STORP | 0809 | 0794 | | | | | | | | | | | |
| STPFL | 0329 | 1021 | 1851 | | | | | | | | | | |
| STROP | 0436 | 1974 | | | | | | | | | | | |
| STSRH | 0448 | | | | | | | | | | | | |
| SUBT | 0542 | 1868 | 2022 | 2022 | | | | | | | | | |
| SUBTA | 2022 | 1946 | | | | | | | | | | | |
| SUMPA | 0233 | | | | | | | | | | | | |
| SYE12 | 0491 | | | | | | | | | | | | |
| SYE25 | 0466 | | | | | | | | | | | | |
| SYMC1 | 0439 | 2000 | | | | | | | | | | | |
| SYMC2 | 0438 | | | | | | | | | | | | |
| SYMCK | 0440 | | | | | | | | | | | | |
| SYMTP | 0304 | | | | | | | | | | | | |
| SYSRE | 1737 | 1375 | 1400 | 1682 | | | | | | | | | |
| SYTSA | 0215 | | | | | | | | | | | | |
| TAP0 | 1825 | 1264 | 1280 | 1711 | 1713 | | | | | | | | |
| TAP1 | 1826 | 0520 | 0522 | 0729 | 0752 | 1311 | 1347 | 1783 | 1785 | | | | |
| TAP2 | 1827 | 0668 | 0679 | 0830 | 0847 | 0883 | 1155 | 1368 | 1383 | 1388 | 1442 | | |
| | 1452 | 1533 | | | | | | | | | | | |
| TAP3 | 1830 | 1358 | 1385 | 1439 | 1450 | 1746 | 1751 | | | | | | |
| TAP4 | 1831 | 0732 | 0744 | 1078 | 1085 | 1089 | 1118 | 1125 | 1266 | 1318 | 1325 | | |
| | 1336 | 1396 | 1633 | 1639 | 1644 | | | | | | | | |
| TAP5 | 1832 | 0730 | 1079 | 1099 | 1119 | 1131 | 1268 | 1327 | 1328 | 1398 | | | |
| TAP6 | 1833 | 1084 | 1096 | 1097 | 1103 | 1124 | 1134 | 1135 | 1137 | | | | |
| TAP7 | 1834 | 0592 | 0628 | 0630 | 0655 | 0659 | 0682 | 0692 | 0712 | 0718 | 0758 | | |

[illegible]

392

1,444,141

392

PAGE 288

CROSS-REFERENCE TABLE

| | | |
|-------|------|------|
| XECA | 1813 | 0903 |
| XFARI | 0431 | |
| XFAR2 | 0435 | |
| XSCRA | 0226 | |
| YC | 0420 | |
| YCOMN | 0692 | 0688 |
| ZAP | 0158 | |
| ZONK | 0496 | |

PAGE 289

MATRIX BLOCK

0501 LST
 0502 20000 ORG 20000B
 0503*
 0504* MATRIX BLOCK
 0505*
 0506* 12001 QUOTE EQU 12001B TABLE OF EQUATES FOR SYNTAX
 0507 12002 COMMA EQU QUOTE+1
 0508 12007 ASSOP EQU QUOTE+6
 0509 12011 PLUS EQU QUOTE+8
 0510 12013 TIMES EQU QUOTE+10
 0511*
 0512*
 0513 20000 000256 ID ABS DETT-* DETERMINANT JUMP TABLE
 0514*
 0515 20001 000100 ABS REDS-* SYNTAX JUMP TABLE
 0516 20002 000041 ABS MATS-*
 0517 20003 000000 NOP ** CHECKSUM WORD **
 0518*
 0519 20004 046501 SYSNT OCT 046501 MA STATEMENT-NAME TABLE
 0520 20005 052242 OCT 052242 T (2)
 0521 20006 051105 OCT 051105 RE
 0522 20007 042111 OCT 042111 DI
 0523 20010 046603 OCT 046603 M (3)
 0524*
 0525 20011 046501 OCT 046501 MA
 0526 20012 052040 OCT 052040 T
 0527 20013 050122 OCT 050122 PR
 0528 20014 044516 OCT 044516 IN
 0529 20015 052204 OCT 052204 T (4)
 0530 20016 046501 OCT 046501 MA
 0531 20017 052040 OCT 052040 T
 0532 20020 051105 OCT 051105 RE
 0533 20021 040504 OCT 040504 AD
 0534 20022 142400 OCT 142400 (5) END
 0535*
 0536 20023 052122 NFOPT OCT 052122 TR NON-FORMULA OPERATOR
 TABLE
 0537 20024 047201 OCT 047201 N (1)
 0538 20025 044516 OCT 044516 IN
 0539 20026 053202 OCT 053202 V (2)
 0540 20027 055105 OCT 055105 ZE
 0541 20030 051203 OCT 051203 R (3)
 0542 20031 041517 OCT 041517 CO
 0543 20032 047204 OCT 047204 N (4)
 0544 20033 044504 OCT 044504 ID
 0545 20034 047305 OCT 047305 N (5) END
 0546*
 0547 20035 042105 FUNCT OCT 042105 DE FUNCTION-NAME TABLE
 0548 20036 052335 OCT 052335 T (29) END
 0549*
 0550 20037 000777 BN777 OCT 777 NULL TABLE
 0551*
 0552 20040 177771 M7 DEC -7
 0553 20041 026000 B26K OCT 26000
 0554 20042 000046 NFOPA DEF NFOPT+NFOPT-ID-ID

PAGE 290

MATRIX BLOCK

0556*

0557* MAT STATEMENT SYNTAX

0558*

0559 20043 021717 MATS LDA NUM GET PAGE INFO

0560 20044 070442 SAR 10

0561 20045 070244 SAL 11

0562 20046 031660 STA T8

0563*

0564 20047 055706 DSZ SBPTR MOVE S-BUFFER POINTER

0565 20050 061055 JSM GNEXT GET FIRST CHARACTER

0566 20051 026220 LDB SUBTA GET SUBSIDIARY TABLE ADDRESS

0567 20052 005660 ADB T8

0568 20053 060537 JSM STSRH DO SEARCH FOR "PRINT" OR "READ"

0569 20054 066107 JMP MATS2 NOT FOUND

0570 20055 045706 ISZ SBPTR

0571 20056 010004 CPA B10K PRINT?

0572 20057 066066 JMP MATS0 YES

0573*

0574* MAT READ SYNTAX

0575*

0576 20060 062227 MATS1 JSM ARRID RECORD ARRAY

0577 20061 062240 JSM MATSB IF SUBSCRIPT, RECORD IT

0578 20062 066064 JMP *+2 NOP

0579 20063 061153 JSM EOLCK END-OF-STATEMENT?

0580 20064 060510 JSM COMCE NO. DEMAND COMMA

0581 20065 066060 JMP MATS1

0582*

0583* MAT PRINT SYNTAX

0584*

0585 20066 062227 MATS0 JSM ARRID RECORD ARRAY

0586 20067 061153 JSM EOLCK END-OF-STATEMENT?

0587 20070 060461 JSM SYMC2 NO. LOOK FOR COMMA OR

SEMICOLON

0588 20071 012001 DEF COMMA-1

0589 20072 164277 JMP E10-2,1 NOT FOUND, ERROR 8

0590 20073 061066 JSM GETSK END-OF-STATEMENT?

0591 20074 066077 JMP *+3 YES

0592 20075 055400 DSZ BADDR NO

0593 20076 066066 JMP MATS0

0594 20077 045706 ISZ SBPTR

0595 20100 164203 JMP ACTSA,1

0596*

0597* REDIM SYNTAX

0598*

0599 20101 062227 REDS JSM ARRID RECORD ARRAY

0600 20102 062240 JSM MATSB RECORD SUBSCRIPT

0601 20103 066105 JMP *+2 NOP

0602 20104 164306 JMP E15,1 NONE FOUND, ERROR 15

0603 20105 060510 JSM COMCE DEMAND COMMA

0604 20106 066101 JMP REDS

407

PAGE 291

MATRIX BLOCK

| | | | | | | |
|-------|------------------|--------|-------|-----|---------|--------------------------------|
| 0606* | | | | | | |
| 0607* | OTHER SYNTAX | | | | | |
| 0608* | | | | | | |
| 0609 | 20107 | 045706 | MATS2 | ISZ | SBPTR | |
| 0610 | 20110 | 061141 | | JSM | LTR+1 | LETTER? |
| 0611 | 20111 | 164321 | | JMP | E25+1,I | NO, ERROR 26 |
| 0612 | 20112 | 021706 | | LDA | SBPTR | SAVE CHARACTER ADDRESS |
| 0613 | 20113 | 031723 | | STA | ARYAD | |
| 0614 | 20114 | 062231 | | JSM | ARRI | RECORD ARRAY |
| 0615 | 20115 | 060463 | | JSM | SYMC1 | LOOK FOR "=" |
| 0616 | 20116 | 012006 | | DEF | ASSOP-1 | |
| 0617 | 20117 | 164325 | | JMP | E30,I | NOT FOUND, ERROR |
| 0618 | 20120 | 061055 | | JSM | GNEXT | |
| 0619 | 20121 | 045706 | | ISZ | SBPTR | |
| 0620 | 20122 | 026042 | | LDB | NFOPA | SEARCH NON-FORMULA |
| | | | | | | OPERATOR TABLE |
| 0621 | 20123 | 005660 | | ADB | T8 | |
| 0622 | 20124 | 060546 | | JSM | STEXP | |
| 0623 | 20125 | 066147 | | JMP | MATS8 | NOT FOUND, MUST BE ARITHMETIC |
| 0624 | 20126 | 045706 | | ISZ | SBPTR | |
| 0625 | 20127 | 070202 | | SAR | 5 | |
| 0626 | 20130 | 050040 | | AND | B37 | RETRIEVE OPCODE |
| 0627 | 20131 | 031661 | | STA | T9 | |
| 0628 | 20132 | 000134 | | ADA | M3 | INV OR TRN? |
| 0629 | 20133 | 070252 | | SAM | MATS3 | YES |
| 0630 | 20134 | 061066 | | JSM | GETSK | NO, END-OF-STATEMENT? |
| 0631 | 20135 | 164203 | | JMP | ACTSA,I | YES |
| 0632 | 20136 | 062240 | | JSM | MATSB | NO, GO RECORD SUBSCRIPT |
| 0633 | 20137 | 065152 | | JMP | NOEOF | |
| 0634* | | | | | | |
| 0635* | INV & TRN SYNTAX | | | | | |
| 0636* | | | | | | |
| 0637 | 20140 | 061055 | MATS3 | JSM | GNEXT | |
| 0638 | 20141 | 060521 | | JSM | LPCKE | DEMAND LEFT PARENTHESIS |
| 0639 | 20142 | 062227 | | JSM | ARRID | RECORD ARRAY |
| 0640 | 20143 | 060534 | | JSM | RPCKE | DEMAND RIGHT PARENTHESIS |
| 0641 | 20144 | 055661 | | DSZ | T9 | TRN? |
| 0642 | 20145 | 065151 | | JMP | EOST | NO |
| 0643 | 20146 | 066205 | | JMP | LHSC | GO CHECK PREVIOUS ARRAY |
| | | | | | | IDENTIFIER |
| 0644* | | | | | | |
| 0645 | 20147 | 061141 | MATS8 | JSM | LTR+1 | LETTER? |
| 0646 | 20150 | 066154 | | JMP | MATS4 | NO, IT MUST BE SCALAR MULTIPLY |
| 0647 | 20151 | 060752 | | JSM | LETCK | LETTER? |
| 0648 | 20152 | 066164 | | JMP | MATS5 | NO, IT MUST BE MAT ARITHMETIC |
| 0649 | 20153 | 164326 | | JMP | E30+1 | YES, ERROR 31 |



PAGE 292

MATRIX BLOCK

0651*
0652* SCALAR MULTIPLY SYNTAX
0653*
0654 20154 060521 MATS4 JSM LPCKE DEMAND LEFT PARENTHESIS
0655 20155 160202 JSM FSCA,I FETCH FORMULA
0656 20156 060534 JSM RPCKE DEMAND RIGHT PARENTHESIS
0657 20157 060463 JSM SYMCI LOOK FOR "***"
0658 20160 012012 DEF TIMES-1
0659 20161 164326 JMP E30+1,I NOT FOUND, ERROR 31
0660 20162 062227 JSM ARRID RECORD ARRAY
0661 20163 065151 JMP EOST
0662*
0663* MAT ARITHMETIC SYNTAX
0664*
0665 20164 055706 MATS5 DSZ SBPTR
0666 20165 062231 JSM ARRI RECORD ARRAY
0667 20166 061153 JSM EOLCK END-OF-STATEMENT?
0668 20167 024020 LDB .3 NO, SEARCH FOR "+" "-" OR "***"
0669 20170 060464 JSM SYMCK
0670 20171 012010 DEF PLUS-1
0671 20172 164326 JMP E30+1,I NOT FOUND, ERROR 31
0672 20173 074742 SBR 16 SET B FOR PFLAG
0673 20174 012041 CPA B26K "***?
0674 20175 066212 JMP MATS7 YES
0675*
0676 20176 035712 MATS6 STB PFLAG SET PFLAG
0677 20177 062227 JSM ARRID RECORD SECOND ARRAY
0678 20200 010031 CPA EOL END-OF-STATEMENT?
0679 20201 066203 JMP *+2 YES
0680 20202 065152 JMP NOEOF NO, ERROR
0681 20203 045712 ISZ PFLAG WAS OPERATOR A "***?
0682 20204 164203 JMP ACTSA,I NO
0683 20205 125723 LHSC LDB ARYAD,I YES, RETRIEVE PREVIOUS ARRAY
0684 20206 074504 SBL 6
0685 20207 015473 CPB TEMP SAME AS LEFT HAND SIDE?
0686 20210 164326 JMP E30+1,I YES, ERROR (IMPROPER ASSIGNMENT)
0687 20211 065151 JMP EOST NO
0688*
0689 20212 121723 MATS7 LDA ARYAD,I RETRIEVE ARRAY IDENTIFIER
0690 20213 070504 SAL 6
0691 20214 024132 LDB M1 SET B FOR PFLAG
0692 20215 011473 CPA TEMP SAME AS LEFT HAND SIDE?
0693 20216 164326 JMP E30+1,I YES, ERROR (IMPROPER ASSIGNMENT)
0694 20217 066176 JMP MATS6 NO

PAGE 293

MATRIX BLOCK

0696 20220 000442 SUBTA DEF SUBTB+SUBTB-ID-ID
 0697 20221 050122 SUBTB OCT 050122 PR
 0698 20222 044516 OCT 044516 IN
 0699 20223 052204 OCT 052204 T (4)
 0700 20224 051105 OCT 051105 RE
 0701 20225 040504 OCT 040504 AD
 0702 20226 142400 OCT 142400 (5) END
 0703*
 0704* SUBROUTINE TO FETCH ARRAY IDENTIFIER AND RECORD IT
 0705*
 0706 20227 061140 ARRID JSM LTR LETTER?
 0707 20230 164321 JMP E25+1,I NO, ERROR 26
 0708 20231 021474 ARRI LDA TEMPI RECORD ARRAY IDENTIFIER
 0709 20232 024020 LDB 3 OF UNKNOWN DIMENSIONS
 0710 20233 060451 JSM STROP+1
 0711 20234 070504 SAL 6
 0712 20235 031473 STA TEMP
 0713 20236 021475 LDA TEMP2 RETRIEVE FOLLOWING CHARACTER
 0714 20237 170402 RET
 0715*
 0716* SUBROUTINE TO FETCH MAT SUBSCRIPT AND RECORD IT
 0717*
 0718 20240 060513 MATSB JSM LPCK LOOK FOR LEFT PARENTHESIS
 0719 20241 064357 JMP RETN2 NOT FOUND, RETURN TO P+2
 0720 20242 020125 LDA LBOP CHANGE (TO I
 0721 20243 131706 STA SBPTR,I
 0722 20244 160202 JSM FSCA,I FETCH SUBSCRIPT
 0723 20245 060504 JSM COMCK LOOK FOR COMMA
 0724 20246 066250 JMP *+2 NOT FOUND
 0725 20247 160202 JSM FSCA,I FOUND, FETCH SECOND SUBSCRIPT
 0726 20250 060534 JSM RPCKE DEMAND RIGHT PARENTHESIS
 0727 20251 055706 DSZ SBPTR
 0728 20252 024123 LDB RBOP CHANGE) TO I
 0729 20253 135706 STB SBPTR,I
 0730 20254 045706 ISZ SBPTR
 0731 20255 065153 JMP EOLCK CHECK FOR END-OF-STATEMENT

PAGE 294

MATRIX BLOCK

| | | | | | | | |
|-------|-------------|--------|-------|-----|---------|----------------------------------|--|
| 0733* | | | | | | | |
| 0734* | DETERMINANT | | | | | | |
| 0735* | | | | | | | |
| 0736 | 20256 | 021416 | DETT | LDA | MODE | | |
| 0737 | 20257 | 070453 | | SAP | DETX | SYNTAX OR EXECUTION? | |
| 0738* | | | | | | | |
| 0739 | 20260 | 023777 | | LDA | OPCODE | GET MAT OPCODE | |
| 0740 | 20261 | 131706 | | STA | SBPTR,I | | |
| 0741 | 20262 | 045706 | | ISZ | SBPTR | | |
| 0742 | 20263 | 061055 | | JSM | GNEXT | | |
| 0743 | 20264 | 060521 | | JSM | LPCKE | DEMAND LEFT PARENTHESIS | |
| 0744 | 20265 | 062227 | | JSM | ARRID | RECORD ARRAY | |
| 0745 | 20266 | 060534 | | JSM | RPCKE | DEMAND RIGHT PARENTHESIS | |
| 0746 | 20267 | 064356 | | JMP | RETN3 | | |
| 0747* | | | | | | | |
| 0748* | | | | | | | |
| 0749 | 20270 | 023726 | DETX | LDA | B1A | SAVE FIRST SEVEN TEMPORARIES | |
| 0750 | 20271 | 024024 | | LDB | .7 | | |
| 0751 | 20272 | 162335 | | JSM | MOVST,I | | |
| 0752 | 20273 | 021412 | | LDA | LSTPT | SET UP FOR DETERMINANT EXECUTION | |
| 0753 | 20274 | 031656 | | STA | T6 | | |
| 0754 | 20275 | 121472 | | LDA | TEMPS,I | | |
| 0755 | 20276 | 045472 | | ISZ | TEMPS | | |
| 0756 | 20277 | 045472 | | ISZ | TEMPS | | |
| 0757 | 20300 | 063541 | | JSM | INFO1 | ESTABLISH INFORMATION VECTOR | |
| 0758 | 20301 | 021642 | | LDA | B1+2 | | |
| 0759 | 20302 | 031645 | | STA | B3+2 | | |
| 0760 | 20303 | 063564 | | JSM | COMPI | EXTRACT DIMENSIONS | |
| 0761 | 20304 | 011653 | | CPA | T3 | | |
| 0762 | 20305 | 066307 | | JMP | *+2 | | |
| 0763 | 20306 | 067120 | | JMP | MERR5 | MATRIX NOT SQUARE | |
| 0764* | | | | | | | |
| 0765 | 20307 | 061201 | | JSM | IMPY | N*N | |
| 0766 | 20310 | 025640 | | LDB | B1 | | |
| 0767 | 20311 | 035643 | | STB | B3 | | |
| 0768 | 20312 | 076110 | | RZB | *+2 | GET # OF TEMPORARY WORDS NEEDED | |
| 0769 | 20313 | 070704 | | SAL | 2 | | |
| 0770 | 20314 | 014012 | | CPB | .1 | | |
| 0771 | 20315 | 070744 | | SAL | 1 | | |
| 0772 | 20316 | 025412 | | LDB | LSTPT | | |
| 0773 | 20317 | 074070 | | SIB | *+1 | | |
| 0774 | 20320 | 035644 | | STB | B3+1 | | |
| 0775 | 20321 | 074017 | | ADA | B | | |
| 0776 | 20322 | 063130 | | JSM | XIN2+1 | GO CALCULATE DETERMINANT | |
| 0777 | 20323 | 066324 | | JMP | *+1 | DET=0 RETURN | |
| 0778 | 20324 | 021656 | | LDA | T6 | | |
| 0779 | 20325 | 002040 | | ADA | M7 | | |
| 0780 | 20326 | 031412 | | STA | LSTPT | | |
| 0781 | 20327 | 070070 | | SIA | *+1 | | |
| 0782 | 20330 | 027726 | | LDB | B1A | RESTORE FIRST SEVEN TEMPORARIES | |
| 0783 | 20331 | 170004 | | XFR | | | |
| 0784 | 20332 | 170004 | | XFR | | | |
| 0785 | 20333 | 023731 | | LDA | DELA | | |
| 0786 | 20334 | 064445 | | JMP | XFAR2+1 | | |
| 0787* | | | | | | | |
| 0788 | 20335 | 011122 | MOVST | DEF | 11122B | | |

PAGE 295

MATRIX BLOCK

0790*
0791* EXECUTE MAT PRINT
0792*
0793 20336 021416 EMATP LDA MODE SAVE MODE FLAG
0794 20337 031656 STA T6
0795 20340 121472 MTP3 LDA TEMPS,I
0796 20341 045472 ISZ TEMPS
0797 20342 063541 JSM INFO1 ESTABLISH INFORMATION VECTOR
0798 20343 121472 LDA TEMPS,I
0799 20344 070442 SAR 10 GET FORMAT INDICATOR
0800 20345 074742 SBR 16
0801 20346 035453 STB PRFLG INDICATE "PRINT"
0802 20347 010020 CPA .3
0803 20350 024020 LDB .3 3 IS ; 0 IS ,
0804 20351 021472 LDA TEMPS
0805 20352 011421 CPA PRADD FALSE ALARM?
0806 20353 074742 SBR 16 YES, IT WAS PART OF NEXT STATEMENT
0807 20354 035655 STB T5
0808 20355 021642 LDA B1+2 SET UP BOUNDS INFO
0809 20356 025642 LDB B1+2
0810 20357 063565 JSM COMPR
0811*
0812 20360 020132 LDA M1
0813 20361 031703 STA PLACE
0814 20362 022442 LDA LN72
0815 20363 031623 STA PEND
0816*
0817 20364 020012 LDA .1 START THE PRINT OPERATION
0818 20365 031646 STA T1
0819 20366 020012 MTP1 LDA .1
0820 20367 031652 STA T2
0821 20370 160170 JSM CLERA,I CLEAR BUFFER
0822 20371 063600 MTP2 JSM GETI1 GET AN ARRAY ELEMENT
0823 20372 070742 SAR 16
0824 20373 031705 STA FLOAT INDICATE FIXED-POINT FORMAT
0825 20374 031416 STA MODE
0826 20375 160234 JSM NMOTA,I
0827 20376 025400 LDB BADDR
0828 20377 015623 CP8 PEND CHECK FOR OVERFLOW
0829 20400 066435 JMP OVFW
0830 20401 070742 SAR 16 SIGNAL NUMERIC (NOT QUOTE) TO FILL ROUTINE
0831 20402 031742 STA XC+2
0832 20403 021655 LDA T5 CHECK THE FORMAT INDICATOR
0833 20404 160235 JSM FILL,I SUPPLY NEEDED BLANKS
0834 20405 160176 JSM ELINA,I OUTPUT ONE FIELD
0835 20406 021442 LDA STPFL CHECK STOP FLAG
0836 20407 010040 CPA .31
0837 20410 066437 JMP MTPX
0838 20411 021652 LDA T2
0839 20412 011654 CPA T4
0840 20413 066416 JMP *+3
0841 20414 045652 ISZ T2
0842 20415 066371 JMP MTP2
0843 20416 061344 JSM CRLF1 DOUBLE-SPACE BETWEEN ROWS
0844 20417 061344 JSM CRLF1

(H13)

400

1,444,141

400

PAGE 296

MATRIX BLOCK

| | | | | | | |
|-------|-------|--------|------|-----|----------------|-----------------------------|
| 0846 | 20420 | 021646 | | LDA | T1 | |
| 0847 | 20421 | 011653 | | CPA | T3 | |
| 0848 | 20422 | 066425 | | JMP | *+3 | |
| 0849 | 20423 | 045646 | | ISZ | T1 | |
| 0850 | 20424 | 066366 | | JMP | MTP1 | |
| 0851 | 20425 | 061344 | | JSM | CRLF1 | EXTRA SPACE AFTER MATRICES |
| 0852 | 20426 | 025472 | | LDB | TEMPS | CHECK FOR MORE ARRAYS TO BE |
| | | | | | | PRINTED |
| 0853 | 20427 | 015421 | | CPB | PRADD | |
| 0854 | 20430 | 066437 | | JMP | MTPX | |
| 0855 | 20431 | 121472 | | LDA | TEMPS,I | |
| 0856 | 20432 | 050115 | | AND | B1777 | CHECK FOR TRAILING FORMAT |
| | | | | | | INDICATOR |
| 0857 | 20433 | 070210 | | SZA | MTPX | |
| 0858 | 20434 | 066340 | | JMP | MTP3 | |
| 0859* | | | | | | |
| 0860 | 20435 | 061344 | OVFW | JSM | CRLF1 | |
| 0861 | 20436 | 066370 | | JMP | MTP2-1 | |
| 0862* | | | | | | |
| 0863 | 20437 | 021656 | MTPX | LDA | T6 | RESTORE MODE |
| 0864 | 20440 | 031416 | | STA | MODE | |
| 0865 | 20441 | 164172 | | JMP | XEC4A,I | |
| 0866* | | | | | | |
| 0867 | 20442 | 003433 | LN72 | DEF | RBUFF+RBUFF+73 | |

PAGE 297

MATRIX BLOCK

```

0869*
0870* EXECUTE MAT READ
0871*
0872 20443 121472 EMATR LDA TEMPS,I
0873 20444 045472 ISZ TEMPS
0874 20445 027730 LDB B3A
0875 20446 063542 JSM INFOV ESTABLISH INFORMATION VECTOR
0876 20447 121472 LDA TEMPS,I
0877 20450 050065 AND OPMSK
0878 20451 010125 CPA LBOP IS IT [ ?
0879 20452 063625 JSM REDIR YES, GO REDIMENSION
0880 20453 063563 JSM COMP2
0881 20454 022462 LDA EMRX GO EXECUTE DATA TRANSFERS
0882 20455 062617 JSM COMM
0883 20456 025472 LDB TEMPS
0884 20457 015421 CPB PRADD
0885 20460 164172 JMP XEC4A,I DONE
0886 20461 066443 JMP EMATR
0887*
0888 20462 062463 EMRX JSM *+1
0889 20463 160256 JSM FDAAA,I
0890*
0891* SUBROUTINE TO STASH AN ARRAY ELEMENT
0892*
0893* ELEMENT IS AN AR2
0894*
0895* ROW & COLUMN VALUES ARE IN T1 AND T2
0896*
0897 20464 027730 STASH LDB B3A FIND TARGET ADDRESS
0898 20465 063705 JSM WHERE
0899 20466 021643 LDA B3 FIND TYPE OF ARRAY
0900 20467 064421 JMP STORE+3
0901*
0902*
0903* EXECUTE REDIM
0904*
0905 20470 121472 EREDM LDA TEMPS,I
0906 20471 045472 ISZ TEMPS
0907 20472 027730 LDB B3A
0908 20473 063542 JSM INFOV ESTABLISH INFORMATION VECTOR
0909 20474 063625 JSM REDIR GO REDIMENSION
0910 20475 025472 LDB TEMPS
0911 20476 015421 CPB PRADD
0912 20477 164172 JMP XEC4A,I DONE
0913 20500 066470 JMP EREDM

```

PAGE 298

MATRIX BLOCK

| | | | | | | |
|-------|-------|--------|-------|-----|---------|--|
| 0915 | 20501 | 121472 | EMATX | LDA | TEMPS,I | GET DESTINATION ARRAY |
| 0916 | 20502 | 045472 | | ISZ | TEMPS | |
| 0917 | 20503 | 027730 | | LDB | B3A | |
| 0918 | 20504 | 063542 | | JSM | INFOV | ESTABLISH INFORMATION VECTOR |
| 0919 | 20505 | 121472 | | LDA | TEMPS,I | |
| 0920 | 20506 | 045472 | | ISZ | TEMPS | |
| 0921 | 20507 | 050115 | | AND | B1777 | |
| 0922 | 20510 | 072310 | | RZA | MATX1 | REGULAR MAT ASSIGNMENT? |
| 0923 | 20511 | 121472 | | LDA | TEMPS,I | NO |
| 0924 | 20512 | 070442 | | SAR | 10 | |
| 0925 | 20513 | 010040 | | CPA | .31 | SCALAR MULTIPLY? |
| 0926 | 20514 | 066531 | | JMP | MATX0 | NO |
| 0927 | 20515 | 066644 | | JMP | ESMY | YES |
| 0928* | | | | | | |
| 0929 | 20516 | 063541 | MATX1 | JSM | INFO1 | ESTABLISH INFORMATION VECTOR |
| 0930 | 20517 | 025472 | | LDB | TEMPS | |
| 0931 | 20520 | 015421 | | CPB | PRADD | STATEMENT EXHAUSTED? |
| 0932 | 20521 | 066605 | | JMP | EMRP | YES, GO DO MAT REPLACEMENT |
| 0933 | 20522 | 121472 | | LDA | TEMPS,I | NO, GET SECOND OPERAND |
| 0934 | 20523 | 027727 | | LDB | B2A | |
| 0935 | 20524 | 063542 | | JSM | INFOV | ESTABLISH INFORMATION VECTOR |
| 0936 | 20525 | 121472 | | LDA | TEMPS,I | THEN GET MAT OPERATOR |
| 0937 | 20526 | 070442 | | SAR | 10 | |
| 0938 | 20527 | 000134 | | ADA | M3 | |
| 0939 | 20530 | 066535 | | JMP | MATJ | |
| 0940* | | | | | | |
| 0941 | 20531 | 121472 | MATX0 | LDA | TEMPS,I | IDENTIFY SPECIAL NON-FORMULA
OPERATOR |
| 0942 | 20532 | 045472 | | ISZ | TEMPS | |
| 0943 | 20533 | 070202 | | SAR | 5 | |
| 0944 | 20534 | 050040 | | AND | .31 | |
| 0945 | 20535 | 002537 | MATJ | ADA | XTBL1 | |
| 0946 | 20536 | 070016 | | EXA | | BRANCH TO PROPER ROUTINE |
| 0947 | 20537 | 066537 | XTBL1 | JMP | * | |
| 0948 | 20540 | 066722 | | JMP | ETRN | TRN |
| 0949 | 20541 | 067110 | | JMP | EINV | INV |
| 0950 | 20542 | 066553 | | JMP | EZER | ZER |
| 0951 | 20543 | 066550 | | JMP | ECON | CON |
| 0952 | 20544 | 066562 | | JMP | EIDN | IDN |
| 0953* | | | | | | |
| 0954 | 20545 | 066671 | | JMP | EADD | + |
| 0955 | 20546 | 066673 | | JMP | ESUB | - |
| 0956 | 20547 | 066760 | | JMP | EMUL | * |

416

PAGE 299

MATRIX BLOCK

0958* EXECUTE ZER AND CON

0959*

| | | | | | | |
|------|-------|--------|------|-----|---------|--------------------------|
| 0960 | 20550 | 063622 | ECON | JSM | REDIM | REDIMENSION IF NECESSARY |
| 0961 | 20551 | 024004 | | LDB | B10K | |
| 0962 | 20552 | 066555 | | JMP | *+3 | |
| 0963 | 20553 | 063622 | EZER | JSM | REDIM | REDIMENSION IF NECESSARY |
| 0964 | 20554 | 074742 | | SBR | 16 | |
| 0965 | 20555 | 061256 | | JSM | CLAR2 | SET UP AR2 |
| 0966 | 20556 | 035755 | | STB | AR2+1 | |
| 0967 | 20557 | 063563 | | JSM | COMP2 | SET UP BOUNDS INFO |
| 0968 | 20560 | 062611 | | JSM | CMR1 | DO THE TRANSFERS |
| 0969 | 20561 | 164172 | | JMP | XEC4A,I | DONE |

0970*

0971* EXECUTE IDN

0972*

| | | | | | | |
|------|-------|--------|------|-----|---------|--------------------------|
| 0973 | 20562 | 063622 | EIDN | JSM | REDIM | REDIMENSION IF NECESSARY |
| 0974 | 20563 | 061256 | | JSM | CLAR2 | SET UP AR2 |
| 0975 | 20564 | 063563 | | JSM | COMP2 | SET UP BOUNDS INFO |
| 0976 | 20565 | 011653 | | CPA | T3 | CHECK FOR SQUARE MATRIX |
| 0977 | 20566 | 066570 | | JMP | *+2 | |
| 0978 | 20567 | 067120 | | JMP | MERR5 | |
| 0979 | 20570 | 062611 | | JSM | CMR1 | TRANSFER THE ZEROS |
| 0980 | 20571 | 024004 | | LDB | B10K | |
| 0981 | 20572 | 035755 | | STB | AR2+1 | |
| 0982 | 20573 | 020012 | | LDA | .1 | TRANSFER THE ONES |
| 0983 | 20574 | 031646 | | STA | T1 | |
| 0984 | 20575 | 031652 | | STA | T2 | |
| 0985 | 20576 | 062464 | EID3 | JSM | STASH | |
| 0986 | 20577 | 021646 | | LDA | T1 | |
| 0987 | 20600 | 011653 | | CPA | T3 | |
| 0988 | 20601 | 164172 | | JMP | XEC4A,I | DONE |
| 0989 | 20602 | 045646 | | ISZ | T1 | |
| 0990 | 20603 | 045652 | | ISZ | T2 | |
| 0991 | 20604 | 066576 | | JMP | EID3 | |

0992*

0993* EXECUTE REPLACEMENT

0994*

| | | | | | | |
|------|-------|--------|------|-----|---------|------------------------------------|
| 0995 | 20605 | 021642 | EMRP | LDA | B1+2 | CHECK FOR COMPATIBLE
DIMENSIONS |
| 0996 | 20606 | 063564 | | JSM | COMP1 | |
| 0997 | 20607 | 062613 | | JSM | CMR4 | DO THE TRANSFERS |
| 0998 | 20610 | 164172 | | JMP | XEC4A,I | |

PAGE 300

MATRIX BLOCK

```

1000*
1001* COMMON ROUTINE TO TRANSFER B3 ← AR2
1002*
1003 20611 022641 CMR1 LDA ERPX
1004 20612 066617 JMP COMM
1005*
1006* COMMON ROUTINE TO TRANSFER B3 ← B1
1007*
1008 20613 021641 CMR4 LDA B1+1 CHECK FOR REDUNDANT TRANSFER
1009 20614 011644 CPA B3+1
1010 20615 170402 RET
1011 20616 022640 LDA ETRX
1012*
1013* COMMON ROUTINE MAT EXECUTION (EXCEPT MULTIPLY, INVERSE, & PRINT)
1014*
1015* LDA PAGE ADDRESS OF SOME ROUTINE
1016* JSM COMM
1017*
1018 20617 031655 COMM STA T5
1019 20620 020012 LDA T1 INITIALIZE OUTER LOOP
1020 20621 031646 STA T1
1021 20622 020012 COM1 LDA T1 INITIALIZE INNER LOOP
1022 20623 031652 STA T2
1023 20624 021655 COM2 LDA T5
1024 20625 070016 EXA JSM TO DESIGNATED ROUTINE
1025 20626 021652 LDA T2
1026 20627 011654 CPA T4
1027 20630 066633 JMP +3
1028 20631 045652 ISZ T2
1029 20632 066624 JMP COM2
1030 20633 021646 LDA T1
1031 20634 011653 CPA T3
1032 20635 170402 RET
1033 20636 045646 ISZ T1
1034 20637 066622 JMP COM1
1035*
1036*
1037* 20640 062642 ETRX JSM +2
1038 20641 062643 ERPX JSM +2
1039 20642 063600 JSM GET11
1040 20643 066464 JMP STASH

```

PAGE 301

MATRIX BLOCK

1042*
1043 EXECUTE SCALAR MULTIPLY
1044*
1045 20644 061217 ESMY JSM FETCH EVALUATE EXPRESSION
1046 20645 045472 ISZ TEMPS
1047 20646 121472 LDA TEMPS,I
1048 20647 063541 JSM INFO1 ESTABLISH INFORMATION VECTOR
1049 20650 020214 LDA AR2A
1050 20651 027734 LDB TF1A
1051 20652 170004 XFR
1052 20653 021642 LDA B1+2 CHECK FOR COMPATIBLE DIMENSIONS
1053 20654 063564 JSM COMP1
1054 20655 022660 LDA ESMX DO THE SCALAR MULTIPLY
1055 20656 062617 JSM COMM
1056 20657 164172 JMP XEC4A,I
1057*
1058 20660 062661 ESMX JSM *+1
1059 20661 063600 JSM GETI1
1060 20662 020214 LDA AR2A
1061 20663 027735 LDB TF2A
1062 20664 170004 XFR
1063 20665 023734 LDA TF1A
1064 20666 027735 LDB TF2A
1065 20667 160220 JSM .FMPI,I
1066 20670 066464 JMP STASH
1067*
1068* EXECUTE ADD AND SUBTRACT
1069*
1070 20671 020216 EADD LDA .FADA
1071 20672 066674 JMP *+2
1072 20673 020217 ESUB LDA .FSBA
1073 20674 031656 STA T6
1074 20675 021642 LDA B1+2 CHECK FOR COMPATIBLE DIMENSIONS
1075 20676 011651 CPA B2+2
1076 20677 066701 JMP *+2
1077 20700 067567 JMP MERR8
1078 20701 063564 JSM COMP1
1079 20702 022705 LDA EASX DO THE ADD OR SUBTRACT
1080 20703 062617 JSM COMM
1081 20704 164172 JMP XEC4A,I
1082*
1083 20705 062706 EASX JSM *+1
1084 20706 063600 JSM GETI1
1085 20707 020214 LDA AR2A
1086 20710 027734 LDB TF1A
1087 20711 170004 XFR
1088 20712 063602 JSM GETI2
1089 20713 020214 LDA AR2A
1090 20714 027735 LDB TF2A
1091 20715 170004 XFR
1092 20716 023734 LDA TF1A
1093 20717 027735 LDB TF2A
1094 20720 161656 JSM T6,I
1095 20721 066464 JMP STASH

419

406

1,444,141

406

PAGE 302

EXECUTE TRANSPOSE

```
1097*
1098* EXECUTE TRANSPOSE
1099*
1100* DOES A(I,J)←A(J,I)
1101*
1102 20722 121472 ETRN LDA TEMP,I
1103 20723 063541 JSM INFO1 ESTABLISH INFORMATION VECTOR
1104 20724 063563 JSM COMP2 CHECK FOR COMPATIBLE DIMENSIONS
1105 20725 031656 STA T6
1106 20726 035655 STB T5
1107 20727 021642 LDA B1+2
1108 20730 025642 LDB B1+2
1109 20731 063565 JSM COMPR
1110 20732 011655 CPA T5
1111 20733 066735 JMP *+2
1112 20734 067567 JMP MERR8
1113 20735 015656 CPB T6
1114 20736 066740 JMP *+2
1115 20737 067567 JMP MERR8
1116 20740 022743 LDA ETNX DO THE TRANSPOSE
1117 20741 062617 JSM COMM
1118 20742 164172 JMP XEC4A,I
1119*
1120 20743 062744 ETNX JSM *+1
1121 20744 063600 JSM GETI1
1122 20745 062747 JSM REVRS
1123 20746 062464 JSM STASH
1124*
1125 20747 021646 REVRS LDA T1 SWAP INDEX POINTERS
1126 20750 025652 LDB T2
1127 20751 031652 STA T2
1128 20752 035646 STB T1
1129 20753 021653 LDA T3
1130 20754 025654 LDB T4
1131 20755 031654 STA T4
1132 20756 035653 STB T3
1133 20757 170402 RET
```

PAGE 303

MATRIX BLOCK

1135*
 1136* EXECUTE MULTIPLY
 1137*
 1138* B3(I,K)=SUM B1(I,P)*B2(P,K) FOR P=1 TO N
 1139*
 1140 20760 021642 EMUL LDA B1+2 CHECK FOR COMPATIBLE DIMENSIONS
 1141 20761 025642 LDB B1+2
 1142 20762 063565 JSM COMPR B1 IS (M,N)
 1143 20763 031656 STA T6 T6=N
 1144 20764 035661 STB T9 T9=M
 1145 20765 021651 LDA B2+2
 1146 20766 025651 LDB B2+2
 1147 20767 063565 JSM COMPR B2 IS (N,L)
 1148 20770 015656 CPB T6
 1149 20771 066773 JMP *+2
 1150 20772 067567 JMP MERR8 N'S NOT SAME
 1151 20773 031656 STA T6 T6=L
 1152 20774 035660 STB T8 SAVE N
 1153 20775 063563 JSM COMP2 B3 IS (M,L)
 1154 20776 015661 CPB T9
 1155 20777 067001 JMP *+2
 1156 21000 067567 JMP MERR8
 1157 21001 011656 CPA T6
 1158 21002 067004 JMP *+2
 1159 21003 067567 JMP MERR8
 1160 21004 061256 JSM CLAR2 ZERO B3
 1161 21005 062611 JSM CMR1
 1162 21006 021660 LDA T8
 1163 21007 031653 STA T3 T3=N
 1164 21010 020012 LDA .1 DO THE MULTIPLY
 1165 21011 031656 STA T6 T6=1
 1166 21012 020012 EML3 LDA .1
 1167 21013 031652 STA T2 T2=K
 1168 21014 020012 EML4 LDA .1
 1169 21015 031646 STA T1 T1=P
 1170 21016 063602 EML5 JSM GETI2 B2(P,K)
 1171 21017 020214 LDA AR2A
 1172 21020 027735 LDB TF2A
 1173 21021 170004 XFR
 1174 21022 021653 LDA T3 SWAP N,L
 1175 21023 025654 LDB T4
 1176 21024 031654 STA T4 T4=N
 1177 21025 035653 STB T3 T3=L
 1178 21026 021656 LDA T6 PERMUTE P,K,I
 1179 21027 025646 LDB T1
 1180 21030 031646 STA T1 T1=I
 1181 21031 021652 LDA T2
 1182 21032 035652 STB T2 T2=P
 1183 21033 031656 STA T6 T6=K
 1184 21034 063600 JSM GETI1 B1(I,P)
 1185 21035 020214 LDA AR2A
 1186 21036 027734 LDB TF1A
 1187 21037 170004 XFR
 1188 21040 023734 LDA TF1A
 1189 21041 027735 LDB TF2A
 1190 21042 160220 JSM FMPI CALL TO MULTIPLY

PAGE 304

MATRIX BLOCK

| | | | | | |
|------|-------|--------|-----|---------|-------------|
| 1191 | 21043 | 020214 | LDA | AR2A | |
| 1192 | 21044 | 027735 | LDB | TF2A | |
| 1193 | 21045 | 170004 | XFR | | |
| 1194 | 21046 | 021653 | LDA | T3 | SWAP L,N |
| 1195 | 21047 | 025654 | LDB | T4 | |
| 1196 | 21050 | 031654 | STA | T4 | T4=L |
| 1197 | 21051 | 035653 | STB | T3 | T3=N |
| 1198 | 21052 | 021652 | LDA | T2 | SWAP P,K |
| 1199 | 21053 | 025656 | LDB | T6 | |
| 1200 | 21054 | 031656 | STA | T6 | T6=P |
| 1201 | 21055 | 035652 | STB | T2 | T2=K |
| 1202 | 21056 | 063604 | JSM | GETI3 | B3(I,K) |
| 1203 | 21057 | 020214 | LDA | AR2A | |
| 1204 | 21060 | 027734 | LDB | TF1A | |
| 1205 | 21061 | 170004 | XFR | | |
| 1206 | 21062 | 023734 | LDA | TF1A | |
| 1207 | 21063 | 027735 | LDB | TF2A | |
| 1208 | 21064 | 160216 | JSM | FADA,I | CALL TO ADD |
| 1209 | 21065 | 062464 | JSM | STASH | |
| 1210 | 21066 | 021646 | LDA | T1 | SWAP I,P |
| 1211 | 21067 | 025656 | LDB | T6 | |
| 1212 | 21070 | 031656 | STA | T6 | T6=I |
| 1213 | 21071 | 035646 | STB | T1 | T1=P |
| 1214 | 21072 | 015653 | SPB | T3 | CHECK P |
| 1215 | 21073 | 067076 | JMP | *+3 | |
| 1216 | 21074 | 045646 | ISZ | T1 | |
| 1217 | 21075 | 067016 | JMP | EML5 | |
| 1218 | 21076 | 021652 | LDA | T2 | |
| 1219 | 21077 | 011654 | CPA | T4 | CHECK K |
| 1220 | 21100 | 067103 | JMP | *+3 | |
| 1221 | 21101 | 045652 | ISZ | T2 | |
| 1222 | 21102 | 067014 | JMP | EML4 | |
| 1223 | 21103 | 021656 | LDA | T6 | |
| 1224 | 21104 | 011661 | CPA | T9 | CHECK I |
| 1225 | 21105 | 164172 | JMP | XEC4A,I | DONE |
| 1226 | 21106 | 045656 | ISZ | T6 | |
| 1227 | 21107 | 067012 | JMP | EML3 | |

PAGE 305

MATRIX BLOCK

```

1229*
1230* EXECUTE INVERSE
1231*
1232*      USES GAUSS-JORDAN WITH ROW INTERCHANGE
1233*
1234  21110  021412  EINV  LDA  LSTPT  SET UP FOR INVERSE EXECUTION
1235  21111  031656      STA  T6
1236  21112  121472      LDA  TEMPS,I
1237  21113  063541      JSM  INFO1  ESTABLISH INFORMATION VECTOR
1238  21114  021642      LDA  B1+2  CHECK FOR COMPATIBLE DIMENSIONS
1239  21115  063564      JSM  COMP1
1240  21116  011653      CPA  T3
1241  21117  067122      JMP  *+3
1242  21120  160205  MERR5 JSM  AERRA,I  MATRIX NOT SQUARE
1243  21121  177676      DEC  -66
1244*
1245  21122  063127      JSM  XIN2  CALL DET/INV EXECUTION
1246  21123  067736      JMP  MER10  SINGULAR MATRIX
1247  21124  021656      LDA  T6
1248  21125  031412      STA  LSTPT  RESTORE LSTPT
1249  21126  164172      JMP  XEC4A,I
1250*
1251*
1252  21127  021412  XIN2  LDA  LSTPT  RESERVE INTERCHANGE VECTOR
1253  21130  031676      STA  ADL
1254  21131  001654      ADA  T4
1255  21132  031412      STA  LSTPT  SAVE NEW LSTPT
1256  21133  060554      JSM  BHSTP+1  CHECK FOR OVERLAP
1257  21134  062613      JSM  CMR4  COPY B1 INTO B3
1258  21135  020020      LDA  ONE  SET DEL = 1.0
1259  21136  027731      LDB  DELA
1260  21137  170004      XFR
1261  21140  020012      LDA  .1  DO THE INVERSE
1262  21141  031653  EN1  STA  T3  T3=K
1263  21142  070137      LDB  A
1264  21143  005676      ADB  ADL
1265  21144  074557      STA  B,I  L(K)=K
1266  21145  031646      STA  T1
1267  21146  031652      STA  T2
1268  21147  063604      JSM  GETI3
1269  21150  020214      LDA  AR2A
1270  21151  027732      LDB  BIGA  BIG=A(K,K)
1271  21152  170004      XFR

```

PAGE 306

MATRIX BLOCK

| | | | | | | |
|------|-------|--------|------|-----|---------|----------------------------|
| 1273 | 21153 | 023732 | EIN3 | LDA | BIGA | SEARCH FOR LARGEST ELEMENT |
| 1274 | 21154 | 027734 | | LDB | TF1A | |
| 1275 | 21155 | 170004 | | XFR | | |
| 1276 | 21156 | 021630 | | LDA | TF1 | |
| 1277 | 21157 | 070131 | | SLA | *+2,C | FORM ABS(BIG) |
| 1278 | 21160 | 031630 | | STA | TF1 | |
| 1279 | 21161 | 063604 | | JSM | GETI3 | |
| 1280 | 21162 | 020214 | | LDA | AR2A | |
| 1281 | 21163 | 027735 | | LDB | TF2A | |
| 1282 | 21164 | 170004 | | XFR | | |
| 1283 | 21165 | 020214 | | LDA | AR2A | |
| 1284 | 21166 | 027733 | | LDB | HLDA | |
| 1285 | 21167 | 170004 | | XFR | | |
| 1286 | 21170 | 021634 | | LDA | TF2 | |
| 1287 | 21171 | 070131 | | SLA | *+2,C | FORM ABS(A(I,K)) |
| 1288 | 21172 | 031634 | | STA | TF2 | |
| 1289 | 21173 | 023734 | | LDA | TF1A | |
| 1290 | 21174 | 027735 | | LDB | TF2A | |
| 1291 | 21175 | 160217 | | JSM | .FSBA,I | ABS(BIG)—ABS(A(I,K)) |
| 1292 | 21176 | 021754 | | LDA | AR2 | |
| 1293 | 21177 | 070411 | | SLA | *+8 | |
| 1294 | 21200 | 023733 | | LDA | HLDA | |
| 1295 | 21201 | 027732 | | LDB | BIGA | |
| 1296 | 21202 | 170004 | | XFR | | BIG=A(I,K) |
| 1297 | 21203 | 021646 | | LDA | T1 | |
| 1298 | 21204 | 025653 | | LDB | T3 | |
| 1299 | 21205 | 005676 | | ADB | ADL | |
| 1300 | 21206 | 074557 | | STA | B,I | L(K)=I |
| 1301 | 21207 | 021646 | | LDA | T1 | CHECK I |
| 1302 | 21210 | 011654 | | CPA | T4 | |
| 1303 | 21211 | 067214 | | JMP | EEIR | |
| 1304 | 21212 | 045646 | | ISZ | T1 | |
| 1305 | 21213 | 067153 | | JMP | EIN3 | |

PAGE 307

MATRIX BLOCK

| | | | | | |
|------|-------|--------|------|-----------|------------------|
| 1307 | 21214 | 025653 | EEIR | LDB T3 | INTERCHANGE ROWS |
| 1308 | 21215 | 005676 | | ADB ADL | |
| 1309 | 21216 | 074517 | | LDA B,I | |
| 1310 | 21217 | 011653 | | CPA T3 | |
| 1311 | 21220 | 067261 | | JMP DCPV | |
| 1312 | 21221 | 031655 | | STA T5 | T5=J |
| 1313 | 21222 | 020012 | | LDA I | |
| 1314 | 21223 | 031652 | | STA T2 | T2=I |
| 1315 | 21224 | 021653 | EIN4 | LDA T3 | |
| 1316 | 21225 | 031646 | | STA T1 | |
| 1317 | 21226 | 063604 | | JSM GETI3 | |
| 1318 | 21227 | 021755 | | LDA AR2+1 | FORM U- |
| 1319 | 21230 | 070250 | | SZA *+5 | |
| 1320 | 21231 | 021754 | | LDA AR2 | |
| 1321 | 21232 | 072111 | | SLA *+2,S | |
| 1322 | 21233 | 070071 | | SLA *+1,C | |
| 1323 | 21234 | 031754 | | STA AR2 | |
| 1324 | 21235 | 020214 | | LDA AR2A | |
| 1325 | 21236 | 027733 | | LDB HLDA | HLD=-A(K,I) |
| 1326 | 21237 | 170004 | | XFR | |
| 1327 | 21240 | 021655 | | LDA T5 | |
| 1328 | 21241 | 031646 | | STA T1 | |
| 1329 | 21242 | 063604 | | JSM GETI3 | |
| 1330 | 21243 | 021653 | | LDA T3 | |
| 1331 | 21244 | 031646 | | STA T1 | |
| 1332 | 21245 | 062464 | | JSM STASH | A(K,I)=A(J,I) |
| 1333 | 21246 | 023733 | | LDA HLDA | |
| 1334 | 21247 | 024214 | | LDB AR2A | |
| 1335 | 21250 | 170004 | | XFR | |
| 1336 | 21251 | 021655 | | LDA T5 | |
| 1337 | 21252 | 031646 | | STA T1 | |
| 1338 | 21253 | 062464 | | JSM STASH | A(J,I)=HLD |
| 1339 | 21254 | 021652 | | LDA T2 | CHECK I |
| 1340 | 21255 | 011654 | | CPA T4 | |
| 1341 | 21256 | 067261 | | JMP DCPV | |
| 1342 | 21257 | 045652 | | ISZ T2 | |
| 1343 | 21260 | 067224 | | JMP EIN4 | |

PAGE 308

MATRIX BLOCK

| | | | | | | |
|------|-------|--------|------|-----|---------|---------------------------------------|
| 1345 | 21261 | 023732 | DCPV | LDA | BIGA | DIVIDE COLUMN BY MINUS PIVOT
VALUE |
| 1346 | 21262 | 027734 | | LDB | TF1A | |
| 1347 | 21263 | 170004 | | XFR | | |
| 1348 | 21264 | 021630 | | LDA | TF1 | |
| 1349 | 21265 | 070131 | | SLA | *+2,C | FORM ABS(BIG) |
| 1350 | 21266 | 031630 | | STA | TF1 | |
| 1351 | 21267 | 023735 | | LDA | TF2A | |
| 1352 | 21270 | 170000 | | CLR | | |
| 1353 | 21271 | 027740 | | LDB | EPSE | |
| 1354 | 21272 | 035634 | | STB | TF2 | |
| 1355 | 21273 | 024004 | | LDB | B10K | |
| 1356 | 21274 | 035635 | | STB | TF2+1 | TOLERANCE = 1E-48 |
| 1357 | 21275 | 023734 | | LDA | TF1A | |
| 1358 | 21276 | 027735 | | LDB | TF2A | |
| 1359 | 21277 | 160217 | | JSM | .FSBA,I | |
| 1360 | 21300 | 021754 | | LDA | AR2 | |
| 1361 | 21301 | 070211 | | SLA | *+4 | |
| 1362 | 21302 | 023731 | | LDA | DELA | |
| 1363 | 21303 | 170000 | | CLR | | |
| 1364 | 21304 | 170402 | | RET | | SINGULAR MATRIX DETECTED |
| 1365 | 21305 | 021653 | | LDA | T3 | |
| 1366 | 21306 | 031652 | | STA | T2 | T2=K |
| 1367 | 21307 | 020012 | | LDA | .1 | |
| 1368 | 21310 | 031646 | | STA | T1 | T1=I |
| 1369 | 21311 | 021653 | EIN6 | LDA | T3 | |
| 1370 | 21312 | 011646 | | CPA | T1 | |
| 1371 | 21313 | 067332 | | JMP | EIN7 | |
| 1372 | 21314 | 063604 | | JSM | GETI3 | |
| 1373 | 21315 | 020214 | | LDA | AR2A | |
| 1374 | 21316 | 027734 | | LDB | TF1A | |
| 1375 | 21317 | 170004 | | XFR | | |
| 1376 | 21320 | 023734 | | LDA | TF1A | |
| 1377 | 21321 | 027732 | | LDB | BIGA | |
| 1378 | 21322 | 160221 | | JSM | .FDVA,I | |
| 1379 | 21323 | 021755 | | LDA | AR2+1 | FORM U- |
| 1380 | 21324 | 070250 | | SZA | *+5 | |
| 1381 | 21325 | 021754 | | LDA | AR2 | |
| 1382 | 21326 | 072111 | | SLA | *+2,S | |
| 1383 | 21327 | 070071 | | SLA | *+1,C | |
| 1384 | 21330 | 031754 | | STA | AR2 | |
| 1385 | 21331 | 062464 | | JSM | STASH | A(I,K)=A(I,K)/RIG |
| 1386 | 21332 | 021646 | EIN7 | LDA | T1 | CHECK I |
| 1387 | 21333 | 011654 | | CPA | T4 | |
| 1388 | 21334 | 067337 | | JMP | REDU | |
| 1389 | 21335 | 045646 | | ISZ | T1 | |
| 1390 | 21336 | 067311 | | JMP | EIN6 | |

PAGE 309

MATRIX BLOCK

| | | | | | |
|------|-------|--------|-------|-------------|--------------------------|
| 1392 | 21337 | 020012 | REDU | LDA .1 | REDUCE MATRIX |
| 1393 | 21340 | 031646 | | STA T1 | T1=I |
| 1394 | 21341 | 031655 | | STA T5 | T5=I |
| 1395 | 21342 | 021653 | EIN8 | LDA T3 | |
| 1396 | 21343 | 011646 | | CPA T1 | |
| 1397 | 21344 | 067411 | | JMP EIN11 | |
| 1398 | 21345 | 031652 | | STA T2 | T2=K |
| 1399 | 21346 | 063604 | | JSM GETI3 | |
| 1400 | 21347 | 020214 | | LDA AR2A | |
| 1401 | 21350 | 027733 | | LDB HLDA | HLD=A(I,K) |
| 1402 | 21351 | 170004 | | XFR | |
| 1403 | 21352 | 020012 | | LDA .1 | |
| 1404 | 21353 | 031652 | | STA T2 | T2=J |
| 1405 | 21354 | 021653 | EIN9 | LDA T3 | |
| 1406 | 21355 | 011652 | | CPA T2 | |
| 1407 | 21356 | 067404 | | JMP EIN10 | |
| 1408 | 21357 | 031646 | | STA T1 | T1=K |
| 1409 | 21360 | 063604 | | JSM GETI3 | |
| 1410 | 21361 | 020214 | | LDA AR2A | |
| 1411 | 21362 | 027734 | | LDB TF1A | |
| 1412 | 21363 | 170004 | | XFR | |
| 1413 | 21364 | 023733 | | LDA HLDA | |
| 1414 | 21365 | 027734 | | LDB TF1A | |
| 1415 | 21366 | 160220 | | JSM FMPIA,I | |
| 1416 | 21367 | 020214 | | LDA AR2A | |
| 1417 | 21370 | 027734 | | LDB TF1A | HLD*A(K,J) |
| 1418 | 21371 | 170004 | | XFR | |
| 1419 | 21372 | 021655 | | LDA T5 | |
| 1420 | 21373 | 031646 | | STA T1 | T1=I |
| 1421 | 21374 | 063604 | | JSM GETI3 | |
| 1422 | 21375 | 020214 | | LDA AR2A | |
| 1423 | 21376 | 027735 | | LDB TF2A | |
| 1424 | 21377 | 170004 | | XFR | |
| 1425 | 21400 | 023734 | | LDA TF1A | |
| 1426 | 21401 | 027735 | | LDB TF2A | |
| 1427 | 21402 | 160216 | | JSM FADA,I | |
| 1428 | 21403 | 062464 | | JSM STASH | A(I,J)=HLD*A(K,J)+A(I,J) |
| 1429 | 21404 | 021652 | EIN10 | LDA T2 | CHECK J |
| 1430 | 21405 | 011654 | | CPA T4 | |
| 1431 | 21406 | 067411 | | JMP *+3 | |
| 1432 | 21407 | 045652 | | ISZ T2 | |
| 1433 | 21410 | 067354 | | JMP EIN9 | |
| 1434 | 21411 | 021646 | EIN11 | LDA T1 | CHECK I |
| 1435 | 21412 | 011654 | | CPA T4 | |
| 1436 | 21413 | 067417 | | JMP *+4 | |
| 1437 | 21414 | 045646 | | ISZ T1 | |
| 1438 | 21415 | 045655 | | ISZ T5 | |
| 1439 | 21416 | 067342 | | JMP EIN8 | |

PAGE 310

MATRIX BLOCK

| | | | | | |
|------|-------|--------|-------|-------------|---------------------|
| 1441 | 21417 | 021653 | | LDA T3 | DIVIDE ROW BY PIVOT |
| 1442 | 21420 | 031646 | | STA T1 | T1=K |
| 1443 | 21421 | 020012 | | LDA .1 | |
| 1444 | 21422 | 031652 | | STA T2 | T2=J |
| 1445 | 21423 | 021653 | EIN12 | LDA T3 | |
| 1446 | 21424 | 011652 | | CPA T2 | |
| 1447 | 21425 | 067436 | | JMP EIN13 | |
| 1448 | 21426 | 063604 | | JSM GETI3 | |
| 1449 | 21427 | 020214 | | LDA AR2A | |
| 1450 | 21430 | 027734 | | LDB TF1A | |
| 1451 | 21431 | 170004 | | XFR | |
| 1452 | 21432 | 023734 | | LDA TF1A | |
| 1453 | 21433 | 027732 | | LDB BIGA | |
| 1454 | 21434 | 160221 | | JSM .FDVA,I | |
| 1455 | 21435 | 062464 | | JSM STASH | A(K,J)=A(K,J)/BIG |
| 1456 | 21436 | 021652 | EIN13 | LDA T2 | CHECK J |
| 1457 | 21437 | 011654 | | CPA T4 | |
| 1458 | 21440 | 067443 | | JMP *+3 | |
| 1459 | 21441 | 045652 | | ISZ T2 | |
| 1460 | 21442 | 067423 | | JMP EIN12 | |
| 1461 | 21443 | 023731 | | LDA DELA | PRODUCT OF PIVOTS |
| 1462 | 21444 | 027732 | | LDB BIGA | |
| 1463 | 21445 | 160220 | | JSM .FMPI,I | |
| 1464 | 21446 | 020214 | | LDA AR2A | |
| 1465 | 21447 | 027731 | | LDB DELA | |
| 1466 | 21450 | 170004 | | XFR | |
| 1467 | 21451 | 023734 | | LDA TF1A | |
| 1468 | 21452 | 170000 | | CLR | |
| 1469 | 21453 | 024014 | | LDB B10K | |
| 1470 | 21454 | 035631 | | STB TF1+1 | |
| 1471 | 21455 | 023734 | | LDA TF1A | |
| 1472 | 21456 | 027732 | | LDB BIGA | |
| 1473 | 21457 | 160221 | | JSM .FDVA,I | |
| 1474 | 21460 | 021653 | | LDA T3 | |
| 1475 | 21461 | 031646 | | STA T1 | |
| 1476 | 21462 | 031652 | | STA T2 | |
| 1477 | 21463 | 062464 | | JSM STASH | A(K,K)=1./BIG |
| 1478 | 21464 | 021653 | | LDA T3 | CHECK K |
| 1479 | 21465 | 011654 | | CPA T4 | |
| 1480 | 21466 | 067471 | | JMP *+3 | |
| 1481 | 21467 | 070070 | SIA | *+1 | |
| 1482 | 21470 | 067141 | | JMP EIN1 | |

PAGE 311

MATRIX BLOCK

| | | | | | | |
|------|-------|--------|-------|-----|-------|-----------------------|
| 1484 | 21471 | 055653 | EIN14 | DSZ | T3 | FINAL ROW INTERCHANGE |
| 1485 | 21472 | 067474 | | JMP | *+2 | |
| 1486 | 21473 | 064357 | | JMP | RETN2 | DONE |
| 1487 | 21474 | 025653 | | LDB | T3 | |
| 1488 | 21475 | 005676 | | ADB | ADL | |
| 1489 | 21476 | 074517 | | LDA | B,I | |
| 1490 | 21477 | 011653 | | CPA | T3 | |
| 1491 | 21500 | 067471 | | JMP | EIN14 | |
| 1492 | 21501 | 031655 | | STA | T5 | T5=L(K) |
| 1493 | 21502 | 020012 | | LDA | .1 | |
| 1494 | 21503 | 031646 | | STA | T1 | T1=J |
| 1495 | 21504 | 021653 | EIN15 | LDA | T3 | |
| 1496 | 21505 | 031652 | | STA | T2 | T2=K |
| 1497 | 21506 | 063604 | | JSM | GETI3 | |
| 1498 | 21507 | 020214 | | LDA | AR2A | |
| 1499 | 21510 | 027733 | | LDB | HLDA | HLD=A(J,K) |
| 1500 | 21511 | 170004 | | XFR | | |
| 1501 | 21512 | 021655 | | LDA | T5 | |
| 1502 | 21513 | 031652 | | STA | T2 | T2=I |
| 1503 | 21514 | 063604 | | JSM | GETI3 | |
| 1504 | 21515 | 021755 | | LDA | AR2+1 | FORM U- |
| 1505 | 21516 | 070250 | | SZA | *+5 | |
| 1506 | 21517 | 021754 | | LDA | AR2 | |
| 1507 | 21520 | 072111 | | SLA | *+2,S | |
| 1508 | 21521 | 070071 | | SLA | *+1,C | |
| 1509 | 21522 | 031754 | | STA | AR2 | |
| 1510 | 21523 | 021653 | | LDA | T3 | |
| 1511 | 21524 | 031652 | | STA | T2 | T2=K |
| 1512 | 21525 | 062464 | | JSM | STASH | A(J,K)=A(J,I) |
| 1513 | 21526 | 021655 | | LDA | T5 | |
| 1514 | 21527 | 031652 | | STA | T2 | T2=I |
| 1515 | 21530 | 023733 | | LDA | HLDA | |
| 1516 | 21531 | 024214 | | LDB | AR2A | |
| 1517 | 21532 | 170004 | | XFR | | |
| 1518 | 21533 | 062464 | | JSM | STASH | A(J,I)=HLD |
| 1519 | 21534 | 021646 | | LDA | T1 | CHECK J |
| 1520 | 21535 | 011654 | | CPA | T4 | |
| 1521 | 21536 | 067471 | | JMP | EIN14 | |
| 1522 | 21537 | 045646 | | ISZ | T1 | |
| 1523 | 21540 | 067504 | | JMP | EIN15 | |

429

416

1,444,141

416

PAGE 312

MATRIX BLOCK

```

1525*
1526* SUBROUTINE TO EXTRACT ARRAY INFORMATION
1527*
1528*   LDA NAME INFORMATION
1529*   LDB INFORMATION VECTOR ADDRESS
1530*   JSM INFOV
1531*
1532 21541 027726 INFO1 LDB B1A ALTERNATE ENTRY FOR B1
1533 21542 035657 INFOV STB T7 TEMPORARY SAVE
1534 21543 050115 AND B1777
1535 21544 160177 JSM SSYMA,I SEARCH SYMBOL TABLE
1536 21545 164340 JMP E401,I ERROR, NOT FOUND (UNDEFINED
                                OPERAND)

1537 21546 074517 LDA B,I GET TYPE INFORMATION
1538 21547 050040 AND 31 ISOLATE TYPE-FIELD
1539 21550 070042 SAR 2 LOOK AT SIZE
1540 21551 131657 STA T7,I AND SAVE IT
1541 21552 045657 ISZ T7
1542 21553 004132 ADB M1
1543 21554 074517 LDA B,I GET VALUE TABLE ADDRESS
1544 21555 131657 STA T7,I AND SAVE AS BASE ADDRESS
1545 21556 045657 ISZ T7
1546 21557 000132 ADA M1
1547 21560 070517 LDA A,I GET ACTUAL DIMENSIONS
1548 21561 131657 STA T7,I AND SAVE THEM
1549 21562 170402 RET

1550*
1551* SUBROUTINE TO EXTRACT AND COMPARE DIMENSIONS OF TWO MATRICES
1552*
1553*   LDA DIMENSION INFORMATION #1
1554*   LDB DIMENSION INFORMATION #2
1555*   JSM COMPR
1556*
1557 21563 021654 COMP2 LDA B3+2
1558 21564 025645 COMP1 LDB B3+2
1559 21565 074057 COMPR CPA B EQUAL?
1560 21566 067571 JMP *+3 YES
1561 21567 160205 MERR8 JSM AERRA,I INCOMPATIBLE DIMENSIONS
1562 21570 177673 DEC -69
1563 21571 074342 SBR 8
1564 21572 074070 SIB *+1
1565 21573 035653 STB T3 SAVE # OF ROWS
1566 21574 050105 AND B377
1567 21575 070070 SIA *+1
1568 21576 031654 STA T4 SAVE # OF COLUMNS
1569 21577 170402 RET

```

PAGE 313

MATRIX BLOCK

1571*
 1572* SUBROUTINE TO OBTAIN AN ARRAY ELEMENT
 1573*
 1574* ELEMENT PLACED IN AR2
 1575*
 1576* ROW & COLUMN VALUES ARE IN T1 AND T2
 1577*
 1578* JSM GETI(N)
 1579*
 1580 21600 027726 GETI1 LDB B1A
 1581 21601 067605 JMP *+4
 1582 21602 027727 GETI2 LDB B2A
 1583 21603 067605 JMP *+2
 1584 21604 027730 GETI3 LDB B3A
 1585 21605 063705 JSM WHERE
 1586 21606 074517 LDA B,I CHECK IF UNDEFINED
 1587 21607 010062 CPA FLGBT
 1588 21610 164337 JMP E40,I ERROR, NOT DEFINED (NO VALUE
 ASSIGNED)
 1589 21611 121657 LDA T7,I FIND TYPE OF ARRAY
 1590 21612 072110 RZA *+2
 1591 21613 067620 JMP GTRL REAL
 1592 21614 010012 CPA .1
 1593 21615 164231 JMP SPLFA,I SPLIT
 1594 21616 074537 LDB B,I
 1595 21617 164227 JMP FXFLA,I INTEGER
 1596 21620 074117 GTRL LDA B
 1597 21621 064445 JMP XFAR2+1



PAGE 314

MATRIX BLOCK

1599*
1600* SUBROUTINE TO REDIMENSION AN ARRAY
1601*
1602 21622 021472 REDIM LDA TEMPS
1603 21623 011421 CPA PRADD
1604 21624 170402 RET
1605 21625 063673 REDIR JSM MCKS GET ROW DIMENSION
1606 21626 035646 STB T1 SAVE IT
1607 21627 024012 LDB 1 LOAD DEFAULT COLUMN COUNT
1608 21630 121472 LDA TEMPS,I HOW MANY DIMENSIONS?
1609 21631 050065 AND OPMSK
1610 21632 010121 CPA B4000
1611 21633 063673 JSM MCKS
1612 21634 045472 ISZ TEMPS
1613 21635 035652 STB T2 SAVE COLUMN DIMENSION
1614 21636 021646 LDA T1 RECALL ROW DIMENSION
1615 21637 061201 JSM IMPY
1616 21640 031657 STA T7 SAVE NEW PRODUCT
1617 21641 025644 LDB B3+1 RECALL FORMAL DIMENSIONS
1618 21642 004133 ADB M2
1619 21643 074517 LDA B,I
1620 21644 070137 LDB A
1621 21645 050105 AND B377
1622 21646 074342 SBR 8
1623 21647 070070 SIA *+1
1624 21650 074070 SIB *+1
1625 21651 061201 JSM IMPY RETURNS WITH FORMAL PRODUCT
1626 21652 070056 CMA
1627 21653 001657 ADA T7 COMPARE WITH NEW PRODUCT
1628 21654 070112 SAM *+2 NEW>OLD?
1629 21655 067703 JMP MERR7 YES, ERROR
1630 21656 021646 LDA T1 NO, ESTABLISH NEW ACTUAL
DIMENSIONS
1631 21657 025652 LDB T2
1632 21660 031653 STA T3 FIRST INTO T3 & T4
1633 21661 035654 STB T4
1634 21662 000132 ADA M1
1635 21663 004132 ADB M1
1636 21664 070404 SAL 8 THEN PACK THE DIMENSIONS
1637 21665 074217 IOR B
1638 21666 025644 LDB B3+1 GET VALUE TABLE ADDRESS
1639 21667 004132 ADB M1
1640 21670 074557 STA B,I STASH PACKED DIMENSIONS
1641 21671 031645 STA B3+2
1642 21672 170402 RET
1643*
1644*
1645 21673 061217 MCKS JSM FETCH EVALUATE SUBSCRIPT EXPRESSION
1646 21674 160225 JSM FLTRA,I CONVERT TO ROUNDED INTEGER
1647 21675 067703 JMP MERR7 INTEGER OVERFLOW
1648 21676 074117 LDA B
1649 21677 000132 ADA M1
1650 21700 070152 SAM *+3 <1?
1651 21701 000157 ADA M256 YES
1652 21702 071412 SAM MCKS-1 >256?
1653 21703 160205 MERR7 JSM AERRA,I NEW DIMENSIONS IMPOSSIBLE
1654 21704 177675 DEC -67

432

PAGE 315

MATRIX BLOCK

1656*
 1657* SUBROUTINE TO COMPUTE ADDRESS OF A MATRIX ELEMENT
 1658*
 1659* ROW & COLUMN VALUES ARE IN T1 AND T2 RESPECTIVELY
 1660*
 1661* LDB INFORMATION VECTOR ADDRESS
 1662* JSM WHERE
 1663*
 1664* RETURNS WITH ADDRESS IN (A) & (B)
 1665*
 1666* ADDRESS=BASE+SIZE*(T4*(T1-1)+(T2-1))
 1667*
 1668 21705 035657 WHERE STB T7 TEMPORARY SAVE
 1669 21706 021646 LDA T1
 1670 21707 000132 ADA M1
 1671 21710 025654 LDB T4
 1672 21711 061201 JSM IMPY RETURNS WITH T4*(T1-1)
 1673 21712 001652 ADA T2
 1674 21713 000132 ADA M1
 1675 21714 125657 LDB T7,I
 1676 21715 076110 RZB *+2
 1677 21716 070704 SAL 2
 1678 21717 014012 CPB 1
 1679 21720 070744 SAL 1
 1680 21721 025657 LDB T7
 1681 21722 074070 SIB *+1
 1682 21723 074417 ADA B,I
 1683 21724 070137 LDB A
 1684 21725 170402 RET

423

420

1,444,141

420

PAGE 316

MATRIX BLOCK

1686*

1687*

MATRIX CONSTANTS

1688*

1689

21726

0001640

B1A

DEF

B1

1690

21727

0001647

B2A

DEF

B2

1691

21730

0001643

B3A

DEF

B3

1692

21731

0001662

DELA

DEF

DEL

1693

21732

0001666

BIGA

DEF

BIG

1694

21733

0001672

HLDA

DEF

HLD

1695

21734

0001630

TF1A

DEF

TF1

1696

21735

0001634

TF2A

DEF

TF2

1697*

1698

21736

160205

MER10

JSM

AERRA,I SINGULAR MATRIX

1699

21737

177674

DEC

-68

1700*

1701

21740

150000

EPSE

OCT

150000

EXPONENT FOR SINGULARITY

TOLERANCE

1702

00025

LEFT

EQU

21766B-*

1703*

1704*

1705

01640

ORG

16408

1706*

1707

01640

0000000

B1

BSS

3

B1 & B3 MUST COME FIRST

1708

01643

0000000

B3

BSS

3

1709

01646

0000000

T1

NOP

1701

01647

0000000

B2

BSS

3

1711

01652

0000000

T2

NOP

1712

01653

0000000

T3

NOP

1713

01654

0000000

T4

NOP

1714

01655

0000000

T5

NOP

1715

01656

0000000

T6

NOP

1716

01657

0000000

T7

NOP

1717

01660

0000000

T8

NOP

1718

01661

0000000

T9

NOP

1719

01662

0000000

DEL

BSS

4

1720

01666

0000000

BIG

BSS

4

1721

01672

0000000

HLD

BSS

4

1722

01676

0000000

ADL

NOP

1723*

1724

01630

ORG

1630B

1725*

1726

01630

0000000

TF1

BSS

4

1727

01634

0000000

TF2

BSS

4

1728*

1729

01640

070053

SAP

*+1

CONVENIENT "NOP" FOR REFERENCE

434

PAGE 317

MATRIX BLOCK

```
1731 21766 ORG 21766B
1732*
1733 21766 176455 ABS EMATR-* MAT READ EXECUTION
1734 21767 176347 ABS EMATP-* MAT PRINT EXECUTION
1735 21770 176500 ABS EREDM-* REDIM EXECUTION
1736 21771 176510 ABS EMATX-* MAT ARITHMETIC EXECUTION
1737 21772 000000 NOP ** CHECKSUM WORD **
1738*
1739 21773 176030 ABS NFOPT-* NON-FORMULA OPERATOR TABLE
1740 21774 176041 ABS FUNCT-* FUNCTION-NAME TABLE
1741 21774 176042 ABS BN777-* COMMAND-NAME TABLE (NULL
TABLE)

1742 21776 176006 ABS SYSNT-* STATEMENT-NAME TABLE
1743 21777 006000 OP COD OCT 6000 MAT OPCODE=3
1744 END
```

* NO ERRORS

485

422

1,444,141

422

PAGE 318

CROSS-REFERENCE TABLE

| | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|
| .1 | 0020 | 0770 | 0817 | 0819 | 0982 | 1019 | 1021 | 1164 | 1166 | 1168 | 1261 |
| | 1313 | 1367 | 1392 | 1403 | 1443 | 1493 | 1592 | 1607 | 1678 | | |
| .10 | 0071 | 0072 | | | | | | | | | |
| .100 | 0070 | | | | | | | | | | |
| .1000 | 0069 | | | | | | | | | | |
| .11 | 0035 | 0043 | | | | | | | | | |
| .12 | 0036 | | | | | | | | | | |
| .13 | 0039 | | | | | | | | | | |
| .15 | 0048 | 0083 | | | | | | | | | |
| .16 | 0049 | 0084 | | | | | | | | | |
| .1E4 | 0068 | | | | | | | | | | |
| .2 | 0027 | | | | | | | | | | |
| .21 | 0050 | | | | | | | | | | |
| .22 | 0051 | 0087 | | | | | | | | | |
| .23 | 0052 | 0088 | | | | | | | | | |
| .28 | 0053 | | | | | | | | | | |
| .3 | 0028 | 0045 | 0668 | 0709 | 0802 | 0803 | | | | | |
| .31 | 0054 | 0090 | 0836 | 0925 | 0944 | 1538 | | | | | |
| .32 | 0055 | | | | | | | | | | |
| .33 | 0056 | | | | | | | | | | |
| .34 | 0057 | 0091 | | | | | | | | | |
| .37 | 0058 | | | | | | | | | | |
| .4 | 0029 | | | | | | | | | | |
| .40 | 0059 | | | | | | | | | | |
| .41 | 0060 | | | | | | | | | | |
| .43 | 0061 | | | | | | | | | | |
| .45 | 0062 | | | | | | | | | | |
| .46 | 0063 | | | | | | | | | | |
| .48 | 0064 | | | | | | | | | | |
| .5 | 0030 | | | | | | | | | | |
| .58 | 0065 | 0094 | | | | | | | | | |
| .6 | 0031 | 0044 | | | | | | | | | |
| .63 | 0066 | | | | | | | | | | |
| .7 | 0032 | 0750 | | | | | | | | | |
| .8 | 0033 | | | | | | | | | | |
| .9 | 0034 | | | | | | | | | | |
| .99 | 0074 | | | | | | | | | | |
| .BUFA | 0180 | | | | | | | | | | |
| .FADA | 0185 | 1070 | 1208 | 1427 | | | | | | | |
| .FDVA | 0188 | 1378 | 1454 | 1473 | | | | | | | |
| .FMFA | 0187 | 1065 | 1190 | 1415 | 1463 | | | | | | |
| .FSBA | 0186 | 1072 | 1291 | 1359 | | | | | | | |
| .FSR | 0190 | | | | | | | | | | |
| .LNUM | 0314 | | | | | | | | | | |
| .NPWR | 0492 | | | | | | | | | | |
| .SLBF | 0181 | | | | | | | | | | |
| A | 0000 | 1263 | 1547 | 1620 | 1683 | | | | | | |
| ACTSA | 0174 | 0595 | 0631 | 0682 | | | | | | | |
| ADL | 1722 | 1253 | 1264 | 1299 | 1308 | 1488 | | | | | |
| ADRSS | 0327 | | | | | | | | | | |
| AERRA | 0176 | 1242 | 1561 | 1653 | 1698 | | | | | | |
| ARI | 0411 | 0182 | | | | | | | | | |
| ARIA | 0182 | | | | | | | | | | |
| AR2 | 0413 | 0183 | 0966 | 0981 | 1292 | 1318 | 1320 | 1323 | 1360 | 1379 | 1381 |
| | 1384 | 1504 | 1506 | 1509 | | | | | | | |

CROSS-REFERENCE TABLE

| | BIGA | 1270 | 1273 | 1295 | 1345 | 1377 | 1453 | 1462 | 1472 |
|--|-------|------|------|------|------|------|------|------|------|
| | BLANK | 0386 | | | | | | | |
| | BN777 | 0550 | 1741 | | | | | | |
| | C5 | 0015 | | | | | | | |
| | C6 | 0011 | 0114 | | | | | | |
| | C7 | 0022 | 0100 | | | | | | |
| | CHRSA | 0219 | | | | | | | |
| | CLAR2 | 0489 | 0965 | 0974 | 1160 | | | | |
| | CLERA | 0160 | 0821 | | | | | | |
| | CLINE | 0316 | | | | | | | |
| | CLPIA | 0217 | | | | | | | |
| | CMRI | 1003 | 0968 | 0979 | 1161 | | | | |
| | CMR4 | 1008 | 0997 | 1257 | | | | | |
| | COM1 | 1021 | 1034 | | | | | | |
| | COM2 | 1023 | 1029 | | | | | | |
| | COMCE | 0434 | 0580 | 0603 | | | | | |
| | COMCK | 0433 | 0723 | | | | | | |
| | COMM | 1018 | 0882 | 1004 | 1055 | 1080 | 1117 | | |
| | COMMA | 0507 | 0588 | | | | | | |
| | COMPI | 1558 | 0760 | 0996 | 1053 | 1078 | 1239 | | |
| | COMP2 | 1557 | 0880 | 0967 | 0975 | 1104 | 1153 | | |
| | COMPR | 1559 | 0810 | 1109 | 1142 | 1147 | | | |
| | CONST | 0482 | | | | | | | |
| | CPLAC | 0331 | | | | | | | |
| | CRCKA | 0228 | | | | | | | |
| | CRLF | 0495 | | | | | | | |
| | CRLF1 | 0496 | 0843 | 0844 | 0851 | 0860 | | | |
| | CRTN | 0037 | 0038 | 0039 | | | | | |
| | CTYPE | 0330 | | | | | | | |
| | DCOMA | 0210 | | | | | | | |
| | DCPV | 1345 | 1311 | 1341 | | | | | |
| | DEL | 1719 | 1692 | | | | | | |
| | DELA | 1692 | 0785 | 1259 | 1362 | 1461 | 1465 | | |
| | DELSA | 0175 | | | | | | | |
| | DETT | 0736 | 0513 | | | | | | |
| | DETX | 0749 | 0737 | | | | | | |
| | DFLAG | 0388 | | | | | | | |
| | DIGCK | 0461 | | | | | | | |
| | DPERR | 0043 | | | | | | | |
| | DRQST | 0317 | | | | | | | |
| | DSTRT | 0303 | | | | | | | |
| | E | 0080 | | | | | | | |
| | E1 | 0235 | | | | | | | |
| | E10 | 0244 | 0589 | | | | | | |
| | E15 | 0249 | 0602 | | | | | | |
| | E20 | 0254 | | | | | | | |
| | E25 | 0259 | 0611 | 0707 | | | | | |
| | E30 | 0264 | 0617 | 0649 | 0659 | 0671 | 0686 | 0693 | |
| | E35 | 0269 | | | | | | | |
| | E40 | 0274 | 1536 | 1588 | | | | | |
| | E45 | 0279 | | | | | | | |
| | E5 | 0239 | | | | | | | |
| | E50 | 0284 | | | | | | | |
| | EADD | 1070 | 0954 | | | | | | |
| | EASX | 1083 | 1079 | | | | | | |
| | ECALA | 0215 | | | | | | | |

438

PAGE 321

CROSS-REFERENCE TABLE

| | | | | | |
|-------|------|------|------|------|------|
| ECON | 0960 | 0951 | | | |
| EEIR | 1307 | 1303 | | | |
| EID3 | 0985 | 0991 | | | |
| EIDN | 0973 | 0952 | | | |
| EIN1 | 1262 | 1482 | | | |
| EIN10 | 1429 | 1407 | | | |
| EIN11 | 1434 | 1397 | | | |
| EIN12 | 1445 | 1460 | | | |
| EIN13 | 1456 | 1447 | | | |
| EIN14 | 1484 | 1491 | 1521 | | |
| EIN15 | 1495 | 1523 | | | |
| EIN3 | 1273 | 1305 | | | |
| EIN4 | 1315 | 1343 | | | |
| EIN6 | 1369 | 1390 | | | |
| EIN7 | 1386 | 1371 | | | |
| EIN8 | 1395 | 1439 | | | |
| EIN9 | 1405 | 1433 | | | |
| EINV | 1234 | 0949 | | | |
| ELINA | 0169 | 0834 | | | |
| EMATP | 0793 | 1734 | | | |
| EMATR | 0872 | 0886 | 1733 | | |
| EMATX | 0915 | 1736 | | | |
| EML3 | 1166 | 1227 | | | |
| EML4 | 1168 | 1222 | | | |
| EML5 | 1170 | 1217 | | | |
| EMRP | 0995 | 0932 | | | |
| EMRX | 0888 | 0881 | | | |
| EMUL | 1140 | 0956 | | | |
| ENDS | 0478 | | | | |
| ENTEA | 0218 | | | | |
| ENTFA | 0189 | | | | |
| EOL | 0038 | 0678 | | | |
| EOLCK | 0481 | 0579 | 0586 | 0667 | 0731 |
| EOST | 0479 | 0642 | 0661 | 0687 | |
| EPSE | 1701 | 1353 | | | |
| EREDM | 0905 | 0913 | 1735 | | |
| ERPX | 1038 | 1003 | | | |
| ERRA | 0177 | | | | |
| ESMX | 1058 | 1054 | | | |
| ESMY | 1045 | 0927 | | | |
| ESUB | 1072 | 0955 | | | |
| ETNX | 1120 | 1116 | | | |
| ETRN | 1102 | 0948 | | | |
| ETRX | 1037 | 1011 | | | |
| EXP | 0399 | 0400 | | | |
| EXPOP | 0118 | | | | |
| EZER | 0963 | 0950 | | | |
| F | 0081 | | | | |
| FDAAA | 0222 | 0889 | | | |
| FETCH | 0486 | 1045 | 1645 | | |
| FILLA | 0201 | 0833 | | | |
| FIRST | 0365 | | | | |
| FIX | 0499 | | | | |
| FLGBT | 0073 | 1587 | | | |
| FLOAT | 0384 | 0824 | | | |
| FLTFA | 0193 | | | | |

439

426

CROSS-REFERENCE TABLE

[illegible]



PAGE 324

CROSS-REFERENCE TABLE

| | | | | | | | |
|-------|------|------|------|------|------|------|------|
| MATS | 0559 | 0516 | | | | | |
| MATS0 | 0585 | 0572 | 0593 | | | | |
| MATS1 | 0576 | 0581 | | | | | |
| MATS2 | 0609 | 0569 | | | | | |
| MATS3 | 0637 | 0629 | | | | | |
| MATS4 | 0654 | 0646 | | | | | |
| MATS5 | 0665 | 0648 | | | | | |
| MATS6 | 0676 | 0694 | | | | | |
| MATS7 | 0689 | 0674 | | | | | |
| MATS8 | 0645 | 0623 | | | | | |
| MATSB | 0718 | 0577 | 0600 | 0632 | | | |
| MATX0 | 0941 | 0926 | | | | | |
| MATX1 | 0929 | 0922 | | | | | |
| MAW | 0294 | | | | | | |
| MAXIN | 0398 | | | | | | |
| MAXSN | 0147 | | | | | | |
| MBIN1 | 0404 | | | | | | |
| MBIN2 | 0405 | | | | | | |
| MBOX1 | 0349 | | | | | | |
| MCKS | 1645 | 1605 | 1611 | 1652 | | | |
| MDIMA | 0208 | | | | | | |
| MER10 | 1698 | 1246 | | | | | |
| MERB | 0443 | | | | | | |
| MERR5 | 1242 | 0763 | 0978 | | | | |
| MERR7 | 1653 | 1629 | 1647 | | | | |
| MERR8 | 1561 | 1077 | 1112 | 1115 | 1150 | 1156 | 1159 |
| MINIT | 0453 | | | | | | |
| MODE | 0305 | 0736 | 0793 | 0825 | 0864 | | |
| MOVST | 0788 | 0751 | | | | | |
| MSFLG | 0387 | | | | | | |
| MTP1 | 0819 | 0850 | | | | | |
| MTP2 | 0822 | 0842 | 0861 | | | | |
| MTP3 | 0795 | 0858 | | | | | |
| MTPX | 0863 | 0837 | 0854 | 0857 | | | |
| MVTOH | 0475 | | | | | | |
| NDISP | 0380 | | | | | | |
| NDWRT | 0494 | | | | | | |
| NEXT1 | 0448 | | | | | | |
| NEXTA | 0465 | | | | | | |
| NEXTL | 0447 | | | | | | |
| NFOPA | 0554 | 0620 | | | | | |
| NFOPT | 0536 | 0554 | 0554 | 1739 | | | |
| NMOTA | 0200 | 0826 | | | | | |
| NOEOF | 0480 | 0633 | 0680 | | | | |
| NPRIN | 0381 | | | | | | |
| NPWRT | 0493 | | | | | | |
| NUM | 0394 | 0395 | 0559 | | | | |
| NUMCA | 0211 | | | | | | |
| NUMOA | 0212 | | | | | | |
| NXTDT | 0304 | | | | | | |
| NXTST | 0403 | | | | | | |
| OBLNK | 0466 | | | | | | |
| ODGIT | 0490 | | | | | | |
| ONE | 0045 | 1258 | | | | | |
| ONEXT | 0467 | | | | | | |
| OPCHA | 0171 | | | | | | |

142

PAGE 325

CROSS-REFERENCE TABLE

| | | | | | | | |
|--------|------|------|------|------|------|------|------|
| OPCOD | 1743 | 0739 | | | | | |
| OPDMK | 0079 | | | | | | |
| OPMSK | 0076 | 0877 | 1609 | | | | |
| OPTYP | 0395 | | | | | | |
| OTPTR | 0293 | | | | | | |
| OUTCR | 0468 | | | | | | |
| OUTIA | 0179 | | | | | | |
| OVCHA | 0216 | | | | | | |
| OVFW | 0860 | 0829 | | | | | |
| PARMA | 0306 | | | | | | |
| PARMN | 0307 | | | | | | |
| PBPTO | 0313 | | | | | | |
| PBPTR | 0310 | | | | | | |
| PBUFF | 0311 | | | | | | |
| PBUFO | 0312 | | | | | | |
| PEND | 0363 | 0815 | 0828 | | | | |
| PEXMA | 0165 | | | | | | |
| PFLAG | 0389 | 0676 | 0681 | | | | |
| PGIN0 | 0476 | | | | | | |
| PGINT | 0456 | | | | | | |
| PI/2 | 0044 | | | | | | |
| PLACE | 0382 | 0813 | | | | | |
| PLUS | 0509 | 0670 | | | | | |
| PMODE | 0329 | | | | | | |
| PPFF | 0364 | | | | | | |
| PRADD | 0308 | 0805 | 0853 | 0884 | 0911 | 0931 | 1603 |
| PRFLG | 0332 | 0801 | | | | | |
| OTCHA | 0230 | | | | | | |
| QTOP | 0109 | 0161 | | | | | |
| QUOTE | 0506 | 0507 | 0508 | 0509 | 0510 | | |
| RBOP | 0117 | 0728 | | | | | |
| RBUFF | 0358 | 0867 | 0867 | | | | |
| RCOUT | 0333 | | | | | | |
| ROYDA | 0166 | | | | | | |
| REDIN | 1602 | 0960 | 0963 | 0973 | | | |
| REDIR | 1605 | 0879 | 0909 | | | | |
| REDS | 0599 | 0515 | 0604 | | | | |
| REDU | 1392 | 1388 | | | | | |
| REED | 0161 | | | | | | |
| RES | 0415 | | | | | | |
| RESB1 | 0451 | | | | | | |
| RESBP | 0452 | | | | | | |
| RET2 | 0419 | | | | | | |
| RETN2 | 0418 | 0719 | 1486 | | | | |
| RETN3 | 0417 | 0746 | | | | | |
| REVR5 | 1125 | 1122 | | | | | |
| RPCK | 0437 | | | | | | |
| RPCKE | 0438 | 0640 | 0656 | 0726 | 0745 | | |
| RPOP | 0115 | | | | | | |
| RSARE | 0292 | | | | | | |
| RSLTE | 0371 | | | | | | |
| RSPTR | 0416 | | | | | | |
| RSTAK | 0414 | | | | | | |
| RSTKA | 0158 | | | | | | |
| RESULT | 0424 | | | | | | |
| RUNA | 0163 | | | | | | |

448

430

1,444,141

430

PAGE 326

CROSS-REFERENCE TABLE

| | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|
| S | 0082 | | | | | | | | | | | |
| SALTA | 0207 | | | | | | | | | | | |
| SAVBP | 0450 | | | | | | | | | | | |
| SAVEA | 0325 | | | | | | | | | | | |
| SAVEB | 0326 | | | | | | | | | | | |
| SBADR | 0392 | | | | | | | | | | | |
| SBPTR | 0385 | 0564 | 0570 | 0594 | 0609 | 0612 | 0619 | 0624 | 0665 | 0721 | 0727 | |
| | 0729 | 0730 | 0740 | 0741 | | | | | | | | |
| SBPUD | 0429 | | | | | | | | | | | |
| SBUFA | 0199 | | | | | | | | | | | |
| SEC | 0320 | | | | | | | | | | | |
| SFLAG | 0391 | | | | | | | | | | | |
| SGNS | 0491 | | | | | | | | | | | |
| SIGN | 0397 | 0398 | | | | | | | | | | |
| SLCOD | 0328 | | | | | | | | | | | |
| SLEND | 0202 | | | | | | | | | | | |
| SLWST | 0445 | | | | | | | | | | | |
| SNFLA | 0197 | | | | | | | | | | | |
| SPLFA | 0196 | 1593 | | | | | | | | | | |
| SSBPT | 0393 | | | | | | | | | | | |
| SSYMA | 0170 | 1535 | | | | | | | | | | |
| ST | 0375 | | | | | | | | | | | |
| START | 0162 | 0007 | | | | | | | | | | |
| STASH | 0897 | 0985 | 1040 | 1066 | 1095 | 1123 | 1209 | 1332 | 1338 | 1385 | 1428 | |
| | 1455 | 1477 | 1512 | 1518 | | | | | | | | |
| STEXP | 0441 | 0622 | | | | | | | | | | |
| STORI | 0426 | | | | | | | | | | | |
| STORE | 0425 | 0900 | | | | | | | | | | |
| STRFL | 0323 | 0835 | | | | | | | | | | |
| STROP | 0428 | 0710 | | | | | | | | | | |
| STSRH | 0440 | 0568 | | | | | | | | | | |
| SUBTA | 0696 | 0566 | | | | | | | | | | |
| SUBTB | 0697 | 0696 | 0696 | | | | | | | | | |
| SUMPA | 0227 | | | | | | | | | | | |
| SYE12 | 0483 | | | | | | | | | | | |
| SYE25 | 0458 | | | | | | | | | | | |
| SYMC1 | 0431 | 0615 | 0657 | | | | | | | | | |
| SYMC2 | 0430 | 0587 | | | | | | | | | | |
| SYMCK | 0432 | 0669 | | | | | | | | | | |
| SYMTP | 0298 | | | | | | | | | | | |
| SYSNT | 0519 | 1742 | | | | | | | | | | |
| SYTSA | 0209 | | | | | | | | | | | |
| T1 | 1709 | 0818 | 0846 | 0849 | 0983 | 0986 | 0989 | 1020 | 1030 | 1033 | 1125 | |
| | 1128 | 1169 | 1179 | 1180 | 1210 | 1213 | 1216 | 1266 | 1297 | 1301 | 1304 | |
| | 1316 | 1328 | 1331 | 1337 | 1368 | 1370 | 1386 | 1389 | 1393 | 1396 | 1408 | |
| | 1420 | 1434 | 1437 | 1442 | 1475 | 1494 | 1519 | 1522 | 1606 | 1614 | 1630 | |
| | 1669 | | | | | | | | | | | |
| T2 | 1711 | 0820 | 0838 | 0841 | 0984 | 0990 | 1022 | 1025 | 1028 | 1126 | 1127 | |
| | 1167 | 1181 | 1182 | 1198 | 1201 | 1218 | 1221 | 1267 | 1314 | 1339 | 1342 | |
| | 1366 | 1398 | 1404 | 1406 | 1429 | 1432 | 1444 | 1446 | 1456 | 1459 | 1476 | |
| | 1496 | 1502 | 1511 | 1514 | 1613 | 1631 | 1673 | | | | | |
| T3 | 1712 | 0761 | 0847 | 0976 | 0987 | 1031 | 1129 | 1132 | 1163 | 1174 | 1177 | |
| | 1194 | 1197 | 1214 | 1240 | 1262 | 1298 | 1307 | 1310 | 1315 | 1330 | 1365 | |
| | 1369 | 1395 | 1405 | 1441 | 1445 | 1474 | 1478 | 1484 | 1487 | 1490 | 1495 | |
| | 1510 | 1565 | 1632 | | | | | | | | | |
| T4 | 1713 | 0839 | 1026 | 1130 | 1131 | 1175 | 1176 | 1195 | 1196 | 1219 | 1254 | |

CROSS-REFERENCE TABLE

| | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|
| T5 | 1302 | 1340 | 1387 | 1430 | 1435 | 1457 | 1479 | 1520 | 1568 | 1633 | 1671 |
| | 1714 | 0807 | 0832 | 1018 | 1023 | 1106 | 1110 | 1312 | 1327 | 1336 | 1394 |
| T6 | 1419 | 1438 | 1492 | 1501 | 1513 | | | | | | |
| | 1715 | 0753 | 0778 | 0794 | 0863 | 1073 | 1094 | 1105 | 1113 | 1143 | 1148 |
| | 1151 | 1157 | 1165 | 1178 | 1183 | 1199 | 1200 | 1211 | 1212 | 1223 | 1226 |
| T7 | 1235 | 1247 | | | | | | | | | |
| | 1716 | 1533 | 1540 | 1541 | 1544 | 1545 | 1548 | 1589 | 1616 | 1627 | 1668 |
| | 1675 | 1680 | | | | | | | | | |
| T8 | 1717 | 0562 | 0567 | 0621 | 1152 | 1162 | | | | | |
| T9 | 1718 | 0627 | 0641 | 1144 | 1154 | 1224 | | | | | |
| TEMP | 0348 | 0349 | 0685 | 0692 | 0712 | | | | | | |
| TEMP1 | 0350 | 0708 | | | | | | | | | |
| TEMP2 | 0351 | 0713 | | | | | | | | | |
| TEMP3 | 0352 | | | | | | | | | | |
| TEMP4 | 0353 | | | | | | | | | | |
| TEMPS | 0347 | 0754 | 0755 | 0756 | 0795 | 0796 | 0798 | 0804 | 0852 | 0855 | 0872 |
| | 0873 | 0876 | 0883 | 0905 | 0906 | 0910 | 0915 | 0916 | 0919 | 0920 | 0923 |
| | 0930 | 0933 | 0936 | 0941 | 0942 | 1046 | 1047 | 1102 | 1236 | 1602 | 1608 |
| | 1612 | | | | | | | | | | |
| TERM | 0362 | | | | | | | | | | |
| TF1 | 1726 | 1276 | 1278 | 1348 | 1350 | 1470 | 1695 | | | | |
| TF1A | 1695 | 1050 | 1063 | 1086 | 1092 | 1186 | 1188 | 1204 | 1206 | 1274 | 1289 |
| | 1346 | 1357 | 1374 | 1376 | 1411 | 1414 | 1417 | 1425 | 1450 | 1452 | 1467 |
| | 1471 | | | | | | | | | | |
| TF2 | 1727 | 1286 | 1288 | 1354 | 1356 | 1696 | | | | | |
| TF2A | 1696 | 1061 | 1064 | 1090 | 1093 | 1172 | 1189 | 1192 | 1207 | 1281 | 1290 |
| | 1351 | 1358 | 1423 | 1426 | | | | | | | |
| TFLAG | 0318 | | | | | | | | | | |
| TFLG1 | 0322 | | | | | | | | | | |
| TIMES | 0510 | 0658 | | | | | | | | | |
| TMPA | 0203 | | | | | | | | | | |
| TMPOP | 0184 | | | | | | | | | | |
| TREGE | 0370 | | | | | | | | | | |
| TSRHA | 0168 | | | | | | | | | | |
| TSTPT | 0300 | | | | | | | | | | |
| TTY | 0497 | | | | | | | | | | |
| TYPFL | 0078 | | | | | | | | | | |
| UFLAG | 0390 | | | | | | | | | | |
| UKPTR | 0309 | | | | | | | | | | |
| UNITS | 0319 | | | | | | | | | | |
| UNSTK | 0446 | | | | | | | | | | |
| UPARA | 0191 | | | | | | | | | | |
| VALUE | 0297 | | | | | | | | | | |
| VAROA | 0214 | | | | | | | | | | |
| WBUFF | 0421 | | | | | | | | | | |
| WHERE | 1668 | 0898 | 1585 | | | | | | | | |
| WIDTH | 0383 | | | | | | | | | | |
| WRITE | 0159 | | | | | | | | | | |
| XC | 0410 | 0831 | | | | | | | | | |
| XEC4A | 0164 | 0865 | 0885 | 0912 | 0969 | 0988 | 0998 | 1056 | 1081 | 1118 | 1225 |
| | 1249 | | | | | | | | | | |
| XEC6A | 0221 | | | | | | | | | | |
| XFAR1 | 0423 | | | | | | | | | | |
| XFAR2 | 0427 | 0786 | 1597 | | | | | | | | |
| XIN2 | 1252 | 0776 | 1245 | | | | | | | | |
| XSCRA | 0220 | | | | | | | | | | |

LMPS

432

1,444,141

432

PAGE 328

CROSS-REFERENCE TABLE

| | | |
|-------|------|------|
| XTBLI | 0947 | 0945 |
| YC | 0412 | |
| ZAP | 0152 | |
| ZONK | 0488 | |

PAGE 329

STRING OPTION BLOCK

0501 LST
 0502*
 0503 22000 ORG 22000B
 0504*
 0505* LINKS FOR THE MAIN SYSTEM AND
 0506* OTHER OPTION BLOCKS
 0507*
 0508*
 0509 22000 066006 IDI JMP MSL MAIN SYSTEM LINK
 0510 22001 066034 JMP SVARO
 0511 22002 067357 JMP GADD
 0512 22003 067421 JMP PSTR
 0513 22004 066610 JMP TRSTR
 0514 22005 067334 JMP FDAI
 0515*
 0516*
 0517*
 0518* EVALUATE A STRING VARIABLE
 0519* FOR THE PRINT ROUTINE
 0520*
 0521* UPON EXIT:
 0522*
 0523* B=STRING STARTING ADDRESS
 0524* A=STRING LENGTH
 0525* RETURN TO P+2
 0526*
 0527 22006 063357 MSL JSM GADD GET STRING ADDRESS
 0528 22007 024133 LDB M2 EVALUATE SOURCE
 0529 22010 063421 JSM PSTR STRING
 0530 22011 035471 STB L2T2
 0531 22012 074117 LDA B PUT COUNT IN A
 0532 22013 025742 LDB TMP2 CHAR POINTER IN B
 0533 22014 064357 JMP RETN2 RETURN TO P+2
 0534*
 0535*
 0536* LET STATEMENT SYNTAX
 0537* LOOK FOR SOURCE STRING=
 0538*
 0539 22015 062340 LETS JSM SAVEP
 0540 22016 062034 JSM SVARO VARIABLE?
 0541 22017 066064 JMP ABORT NO
 0542 22020 066064 JMP ABORT NUMERIC
 0543 22021 060463 JSM SYMCI STRING
 0544 22022 012020 OCT 12020 =?
 0545 22023 164325 JMP E30,1 No
 0546*
 0547*
 0548*
 0549* LOOK FOR A DESTINATION STRING
 0550*
 0551*
 0552 22024 062034 LTSYM JSM SVARO VARIABLE?
 0553 22025 066030 JMP QUOCK NO, LOOK FOR QUOTE FEILD
 0554 22026 164312 JMP E20-1,1 NUMERIC
 0555 22027 065151 JMP EOST YES, DEMAND END
 0556 22030 045706 QUOCK ISZ SBPTR

447

434

1,444,141

434

PAGE 330

STRING OPTION BLOCK

| | | | | | |
|-------|-------|--------|-----|---------|-----------------------|
| 0557 | 22031 | 160245 | JSM | GTSTA,I | GET A STRING CONSTANT |
| 0558 | 22032 | 065151 | JMP | EOST | |
| 0559* | | | | | |
| 0560* | | | | | |

PAGE 331

STRING OPTION BLOCK

| | | | | | |
|-------|-------|--------|-------|-------------------|-------------------------|
| 0562* | | | | | |
| 0563* | | | | | |
| 0564* | | | SVARO | SVARO SUBROUTINE | |
| 0565* | | | | | |
| 0566* | | | | EXIT CONDITIONS: | |
| 0567* | | | | | |
| 0568* | | | | RETURN TO: | |
| 0569* | | | P+1 | NO VARIABLE FOUND | |
| 0570* | | | P+2 | NUMERIC VARIABLE | |
| 0571* | | | P+3 | STRING VARIABLE | |
| 0572* | | | | | |
| 0573 | 22033 | 062340 | JSM | SAVEP | SAVE PAGE |
| 0574 | 22034 | 061140 | JSM | LTR | LETTER? |
| 0575 | 22035 | 170402 | RET | | NO |
| 0576 | 22036 | 010077 | CPA | B44 | \$? |
| 0577 | 22037 | 066041 | JMP | *+2 | YES |
| 0578 | 22040 | 064357 | JMP | RETN2 | NO |
| 0579 | 22041 | 045644 | ISZ | ST+4 | SET STRING OCCURED FLAG |
| 0580 | 22042 | 163610 | JSM | STR12,I | STORE OPERAND |
| 0581 | 22043 | 060513 | JSM | LPCK | (?) |
| 0582 | 22044 | 064356 | JMP | RETN3 | NO |
| 0583 | 22045 | 020125 | LDA | LBOP | |
| 0584 | 22046 | 131706 | STA | SBPTR,I | RECORD RIGHT BRACKET |
| 0585 | 22047 | 020026 | LDA | .9 | |
| 0586 | 22050 | 031710 | STA | MSFLG | ENABLE LOGICL= |
| 0587 | 22051 | 160202 | JSM | FSCA,I | GET FIRST INDEX |
| 0588 | 22052 | 060463 | JSM | SYMC1 | LOOK FOR A COMMA |
| 0589 | 22053 | 012001 | OCT | 12001 | |
| 0590 | 22054 | 066056 | JMP | *+2 | NOT FOUND |
| 0591 | 22055 | 160202 | JSM | FSCA,I | FOUND |
| 0592 | 22056 | 060534 | JSM | RPCKE | DEMAND RIGHT PAREN |
| 0593 | 22057 | 024123 | LDB | RBOP | |
| 0594 | 22060 | 055706 | DSZ | SBPTR | |
| 0595 | 22061 | 135706 | STB | SBPTR,I | RECORD A RIGHT BRACKET |
| 0596 | 22062 | 045706 | ISZ | SBPTR | |
| 0597 | 22063 | 064356 | JMP | RETN3 | YES |

448

PAGE 332

STRING OPTION BLOCK

0599*

0600*

0601*

0602*

0603*

0604*

0605

0606

0607

0608

0609

0610

0611

0612

0613

0614

0615

0616

0617

0618

0619

0620

0621

0622

0623

0624

0625

0626

0627

0628*

0629*

0630*

0631

0632*

0633

0634

0635

0636

0637

0638

0639

0640

0641

0642

0643

0644

0645

0646

0647

0648

0649

0650

0651

0652

0653

0654

THIS ROUTINE RETURNS SYNTAX CONTROL TO OTHER
OPTION BLOCKS AND THE MAIN SYSTEM

22064

061064

ABORT

JSM

GFRST+2

22065

061055

AB3

JSM

GNEXT

22066

025416

LDB

MODE

22067

074151

SLB

*+3

22070

024233

LDB

SBUFA

22071

066077

JMP

AB1

22072

060750

JSM

DIGCK

22073

066075

JMP

*+2

22074

066065

JMP

ABORT+1

22075

024233

LDB

SBUFA

22076

074070

SIB

*+1

22077

035706

AB1

STB

SBPTR

22100

024132

LDB

M1

22101

035473

STB

TEMP

22102

025640

LDB

ST

22103

035717

STB

NUM

22104

074742

SBR

16

22105

035714

STB

SFLAG

22106

035711

STB

DFLAG

22107

015643

CPB

ST+3

22110

070742

SAR

16

22111

163602

JSM

SALT,I

22112

066112

JMP

*

22113

167601

JMP

STMTA,I

22114

062033

IFSYN

JSM

SVARO-1

22115

066064

JMP

ABORT

22116

066064

JMP

ABORT

22117

026146

LDB

GTEA

22120

005640

ADB

ST

22121

005640

ADB

ST

22122

060546

JSM

STEXP

22123

066143

JMP

IFERR

22124

045706

ISZ

SBPTR

22125

070742

SAR

16

22126

131706

STA

SBPTR,I

22127

062034

IF1

JSM

SVARO

22130

066133

JMP

IF3

22131

066143

JMP

IFERR

22132

066134

JMP

IF2

22133

160245

IF3

JSM

GTSTA,I

22134

026145

IF2

LDB

THENA

22135

005640

ADB

ST

22136

005640

ADB

ST

22137

060546

JSM

STEXP

22140

164317

JMP

E20+4,I

22141

060664

JSM

LNUM

22142

060664

JSM

LNUM

22143

060664

JSM

LNUM

22144

060664

JSM

LNUM

22145

060664

JSM

LNUM

22146

060664

JSM

LNUM

22147

060664

JSM

LNUM

22148

060664

JSM

LNUM

22149

060664

JSM

LNUM

22150

060664

JSM

LNUM

22151

060664

JSM

LNUM

22152

060664

JSM

LNUM

22153

060664

2449

436

1,444,141

436

PAGE 333

STRING OPTION BLOCK

| | | | | | | |
|-------|-------|--------|--------------------------|-----|-----------------|----------------------|
| 0655 | 22142 | 065151 | | JMP | EOST | |
| 0656 | 22143 | 160205 | IFERR | JSM | AERRA,I | STRING IFERROR |
| 0657 | 22144 | 177672 | | DEC | -70 | |
| 0658 | 22145 | 174722 | THENA | ABS | THEN+THEN-ID-ID | |
| 0659 | 22146 | 174727 | GTEA | ABS | GTE+GTE+1-ID-ID | |
| 0660* | | | | | | |
| 0661* | | | | | | |
| 0662* | | | | | | |
| 0663* | | | | | | |
| 0664* | | | | | | |
| | | | IMPLIED DISP, LET SYNTAX | | | |
| 0665 | 22147 | 062340 | IMPLT | JSN | SAVEP | |
| 0666 | 22150 | 031643 | | STA | ST+3 | SET IMPLIED LET FLAG |
| 0667 | 22151 | 055400 | | DSZ | BADDR | |
| 0668 | 22152 | 061055 | | JSM | GNEXT | |
| 0669 | 22153 | 055400 | | DSZ | BADDR | |
| 0670 | 22154 | 060752 | | JSM | LETCK | LETTER? |
| 0671 | 22155 | 066167 | | JMP | ILET1+1 | NO, TRY IMPLIED DISP |
| 0672 | 22156 | 060626 | | JSM | SAVBP | SAVE POINTERS |
| 0673 | 22157 | 062034 | | JSM | SVARO | VARIABLE? |
| 0674 | 22160 | 066166 | | JMP | ILET1 | NO |
| 0675 | 22161 | 066166 | | JMP | ILET1 | NUMERIC |
| 0676 | 22162 | 060463 | | JSM | SYMC1 | STRING |
| 0677 | 22163 | 012020 | | OCT | 12020 | =? |
| 0678 | 22164 | 066166 | | JMP | ILET1 | NO |
| 0679 | 22165 | 066024 | | JMP | LTSYM | YES |
| 0680 | 22166 | 060640 | ILET1 | JSM | RESBP | RESTORE POINTERS |
| 0681 | 22167 | 024123 | | LDB | PRTOP | STORE |
| 0682 | 22170 | 055706 | | DSZ | SPBTR | CODE FOR |
| 0683 | 22171 | 135706 | | STB | SPBTR,I | IMPLIED LET |
| 0684 | 22172 | 045706 | | ISZ | SPBTR | |
| 0685 | 22173 | 025416 | | LDB | MODE | |
| 0686 | 22174 | 074111 | | SLB | NSTMT | SKIP IF PROG MODE |
| 0687 | 22175 | 062227 | | JSM | PRNTS | PRINT STATEMENT? |
| 0688 | 22176 | 066200 | NSTMT | JMP | ILET2 | NO |
| 0689 | 22177 | 066213 | | JMP | PRIN1 | YES |
| 0690 | 22200 | 025644 | ILET2 | LDB | ST+4 | STRING |
| 0691 | 22201 | 074110 | | SZB | *+2 | OCCURRED? |
| 0692 | 22202 | 164274 | | JMP | E5,I | YES |
| 0693 | 22203 | 066064 | | JMP | ABORT | |
| 0694 | 00123 | | PRTOP | EQU | B12K | |
| 0695* | | | | | | |
| 0696* | | | | | | |

450

PAGE 334

STRING OPTION BLOCK

0698*

0699*

0700*

0701*

0702*

WRITE STATEMENT SYNTAX

| | | | | | | |
|-------|-------|--------|-------|-----|----------|-----------------------|
| 0703 | 22204 | 160265 | WRTS | JSM | IOCHEA,I | GET DEVICE AND FORMAT |
| 0704 | 22205 | 060534 | | JSM | RPCKE | GET RIGHT PAREN |
| 0705 | 22206 | 055400 | | DSZ | BADDR | |
| 0706 | 22207 | 055706 | | DSZ | SBPTR | |
| 0707* | | | | | | |
| 0708* | | | | | | |
| 0709 | 22210 | 062340 | PRINS | JSM | SAVEP | |
| 0710 | 22211 | 062227 | | JSM | PRNTS | PRINT SYNTAX? |
| 0711 | 22212 | 066216 | | JMP | PRIN2 | |
| 0712 | 22213 | 025644 | PRIN1 | LDB | ST+4 | YES, STRING OCCURRED? |
| 0713 | 22214 | 074250 | | SZB | ABADD | NO |
| 0714 | 22215 | 065151 | | JMP | EOST | YES |
| 0715 | 22216 | 025644 | PRIN2 | LOB | ST+4 | |
| 0716 | 22217 | 074110 | | SZB | *+2 | STRING OCCURRED? |
| 0717 | 22220 | 164277 | | JMP | E5+3,I | YES |
| 0718 | 22221 | 066064 | ABADD | JMP | ABORT | NO |
| 0719* | | | | | | |
| 0720* | | | | | | |
| 0721* | | | | | | |

PRINT STATEMENT SYNTAX

0722*

0723*

0724*

0725*

0726*

0727

0728

0729

0730

0731

0732

0733

0734

0735

0736

0737

0738

0739

0740

0741

0742

0743

0744

0745

0746

0747

0748

0749

0750

0751

0752

0753

| | | | | | |
|-------|--------|-------|-----|---------|---------------------------|
| 22222 | 010043 | PRSYN | CPA | B42 | QUOTE FIELD? |
| 22223 | 066237 | | JMP | PR2+4 | YES |
| 22224 | 060461 | | JSM | SYMC2 | NO |
| 22225 | 012001 | | OCT | 12001 | COMMA OR SEMICOLON? |
| 22226 | 066233 | | JMP | PR2 | NO |
| 22227 | 024132 | PRNTS | LDB | M1 | |
| 22230 | 035712 | | STB | PFLAG | ENABLE FORMULA AND TAB |
| 22231 | 061066 | | JSM | GETSK | |
| 22232 | 066300 | | JMP | PR1 | END OF LINE |
| 22233 | 010043 | PR2 | CPA | B42 | QUOTE? |
| 22234 | 066236 | | JMP | *+2 | YES |
| 22235 | 066247 | | JMP | PR3 | NO |
| 22236 | 045706 | | ISZ | SBPTR | |
| 22237 | 160245 | | JSM | GTSTA,I | RECORD STRING CONSTANT |
| 22240 | 010031 | | CPA | EOL | EOL? |
| 22241 | 064357 | | JMP | RETN2 | YES |
| 22242 | 074742 | | SBR | 16 | |
| 22243 | 135706 | | STB | SBPTR,I | CLEAR OUT NEXT WORD |
| 22244 | 024132 | | LDB | M1 | |
| 22245 | 035712 | | STB | PFLAG | |
| 22246 | 066222 | | JMP | PRSYN | |
| 22247 | 045712 | PR3 | ISZ | PFLAG | TAB OR FORMULA PERMITTED? |
| 22250 | 170402 | | RET | | |
| 22251 | 027603 | | LDB | TABA | |
| 22252 | 160175 | | JSM | TSRHA,I | TAB? |
| 22253 | 066263 | | JMP | PR4 | NO |
| 22254 | 070544 | | SAL | 5 | YES |

PAGE 335

STRING OPTION BLOCK

| | | | | | | |
|------|-------|--------|-----|-----|---------|--------------------------------|
| 0754 | 22255 | 040067 | | IOR | TYPFL | SET PRE-DEFINED FUNCTION FLAGS |
| 0755 | 22256 | 060454 | | JSM | SBPUD-2 | STORE |
| 0756 | 22257 | 024123 | | LDB | RBOP | |
| 0757 | 22260 | 135706 | | STB | SBPTR,I | |
| 0758 | 22261 | 160202 | | JSM | FSCA,I | GET TAB FORMULA |
| 0759 | 22262 | 066275 | | JMP | PR5 | |
| 0760 | 22263 | 055400 | PR4 | DSZ | BADDR | BACK UP POINTER |
| 0761 | 22264 | 021400 | | LDA | BADDR | |
| 0762 | 22265 | 031645 | | STA | ST+5 | SAVE BADDRR |
| 0763 | 22266 | 062034 | | JSM | SVARO | VARIABLE? |
| 0764 | 22267 | 066272 | | JMP | PR6 | NO |
| 0765 | 22270 | 066272 | | JMP | *+2 | NUMERIC |
| 0766 | 22271 | 066275 | | JMP | PR5 | STRING |
| 0767 | 22272 | 021645 | PR6 | LDA | ST+5 | |
| 0768 | 22273 | 031400 | | STA | BADDR | RESTORE BADDR |
| 0769 | 22274 | 160202 | | JSM | FSCA,I | FETCH FORMULA |
| 0770 | 22275 | 010031 | PR5 | CPA | EOL | EOL? |
| 0771 | 22276 | 064357 | | JMP | RETN2 | |
| 0772 | 22277 | 066222 | | JMP | PRSYN | NO |
| 0773 | 22300 | 045706 | PR1 | ISZ | SBPTR | |
| 0774 | 22301 | 064357 | | JMP | RETN2 | |

0775*

0776*

0777*

READ AND INPUT STATEMENT SYNTAX

0778*

| | | | | | | |
|------|-------|--------|-------|-----|---------|------------------|
| 0779 | 22302 | 062340 | READS | JSM | SAVEP | |
| 0780 | 22303 | 062034 | | JSM | SVARO | VARIABLE? |
| 0781 | 22304 | 066315 | | JMP | READ3 | |
| 0782 | 22305 | 066311 | | JMP | READ1 | NUMERIC |
| 0783 | 22306 | 060504 | READ2 | JSM | COMCK | COMMA? |
| 0784 | 22307 | 066213 | | JMP | PRIN1 | NO |
| 0785 | 22310 | 066303 | | JMP | READS+1 | YES |
| 0786 | 22311 | 163600 | READ1 | JSM | VAR1,I | |
| 0787 | 22312 | 066312 | | JMP | * | CONSTANT STORAGE |
| 0788 | 22313 | 066306 | | JMP | READ2 | |
| 0789 | 22314 | 066306 | | JMP | READ2 | |
| 0790 | 22315 | 025644 | READ3 | LDB | ST+4 | |
| 0791 | 22316 | 074110 | | SZB | *+2 | STRING OCCURRED? |
| 0792 | 22317 | 164321 | | JMP | E25+1,I | YES |
| 0793 | 22320 | 066064 | | JMP | ABORT | NO |

0794*

0795*

0796*

DATA STATEMENT SYNTAX

0797*

| | | | | | | |
|------|-------|--------|-------|-----|---------|-------------------|
| 0798 | 22321 | 062340 | DATAS | JSM | SAVEP | |
| 0799 | 22322 | 045706 | | ISZ | SBPTR | |
| 0800 | 22323 | 061156 | | JSM | CONST | LOOK FOR A NUMBER |
| 0801 | 22324 | 066331 | | JMP | DATA1 | NOT FOUND |
| 0802 | 22325 | 160244 | | JSM | NUMOA,I | FOUND, STORE |
| 0803 | 22326 | 060504 | DATA2 | JSM | COMCK | COMMA? |
| 0804 | 22327 | 066213 | | JMP | PRIN1 | NO |
| 0805 | 22330 | 066322 | | JMP | DATAS+1 | |
| 0806 | 22331 | 045644 | DATA1 | ISZ | ST+4 | |
| 0807 | 22332 | 010043 | | CPA | B42 | QUOTE? |
| 0808 | 22333 | 066335 | | JMP | *+2 | YES |
| 0809 | 22334 | 164316 | | JMP | E20+3,I | NO |

452

PAGE 336

STRING OPTION BLOCK

| | | | | | | |
|---------------------------|-------|--------|-------|-----|---------|----------------------------|
| 0810 | 22335 | 160245 | | JSM | GTSTA,I | STRING CONSTANT? |
| 0811 | 22336 | 066326 | | JMP | DATA2 | YES |
| 0812 | 22337 | 066064 | | JMP | ABORT | NO, ERROR |
| 0813* | | | | | | |
| 0814* | | | | | | |
| 0815 | 22340 | 021717 | SAVEP | LDA | NUM | |
| 0816 | 22341 | 031640 | | STA | ST | |
| 0817 | 22342 | 031643 | | STA | ST+3 | RESET IMPLIED LET FLAG |
| 0818 | 22343 | 070742 | PTR2 | SAR | 16 | |
| 0819 | 22344 | 031644 | | STA | ST+4 | RESET STRING OCCURRED FLAG |
| 0820 | 22345 | 170402 | | RET | | |
| 0821* | | | | | | |
| 0822* | | | | | | |
| 0823* | | | | | | |
| NON-FORMULA OPERATORS | | | | | | |
| 0824 | 22346 | 046105 | LEN | ASC | 1,LE | |
| 0825 | 22347 | 047215 | | OCT | 47215 | N, OPCODE 15 |
| 0826 | 22350 | 052100 | THEN | ASC | 2,THEN | |
| | 22351 | 042516 | | | | |
| 0827 | 22352 | 150074 | GTE | OCT | 150074 | 20,< |
| 0828 | 22353 | 036602 | | OCT | 36602 | =,2 |
| 0829 | 22354 | 037075 | | ASC | 1,>= | |
| 0830 | 22355 | 101443 | | OCT | 101443 | 3,# |
| 0831 | 22356 | 102074 | | OCT | 102074 | 4,< |
| 0832 | 22357 | 037204 | | OCT | 37204 | >,4 |
| 0833 | 22360 | 036605 | | OCT | 36605 | =,5 |
| 0834 | 22361 | 036206 | | OCT | 36206 | <,6 |
| 0835 | 22362 | 037307 | | OCT | 37307 | >,7 |
| 0836* | | | | | | |
| 0837* | | | | | | |
| 0838* | | | | | | |
| EXECUTE LEN STATEMENT | | | | | | |
| 0839* | | | | | | |
| 0840* | | | | | | |
| 0841 | 22363 | 062450 | ELEN | JSM | CKMOD | |
| 0842 | 22364 | 160227 | ELEN1 | JSM | FXFLA,I | PUT NUMBER IN AR2 |
| 0843 | 22365 | 045472 | ELEN2 | ISZ | TEMPS | |
| 0844 | 22366 | 170402 | | RET | | |
| 0845* | | | | | | |
| 0846* | | | | | | |
| CHECK LEN FUNCTION SYNTAX | | | | | | |
| 0847* | | | | | | |
| 0848* | | | | | | |
| 0849 | 22367 | 062536 | LENS | JSM | GVAR | GET FIRST PARAMETER |
| 0850 | 22370 | 066446 | | JMP | LENER | |
| 0851 | 22371 | 066446 | | JMP | LENER | NO |
| 0852 | 22372 | 025646 | LENS1 | LDB | PTR2 | |
| 0853 | 22373 | 135472 | | STB | TEMPS,I | FLAG FOR FSC |
| 0845 | 22374 | 060534 | | JSM | RPCKE |)? |
| 0855 | 22375 | 025651 | | LDB | TEMP6 | RESTORED STRING |
| 0856 | 22376 | 035644 | | SIB | ST+4 | OCCURRED FLG |
| 0857 | 22377 | 064356 | | JMP | RETN3 | YES |
| 0858 | 22400 | 045706 | LENS2 | ISZ | SBPTR | |
| 0859 | 22401 | 160245 | | JSM | GTSTA,I | SEEK STRING CONSTANT |
| 0860 | 22402 | 066372 | | JMP | LENS1 | |
| 0861* | | | | | | |
| 0862* | | | | | | |
| 0863 | 22403 | 021416 | EINDX | LDA | MODE | |
| 0864 | 22404 | 070113 | | SAP | *+2 | SYNTAX MODE? |

454

PAGE 338

STRING OPTION BLOCK

| | | | | | | |
|-------|-------|--------|-------|-----|---------|----------------------------|
| 0921 | 22470 | 066400 | | JMP | LENS2 | NO, LOOK FOR QUOTE FEILD |
| 0922 | 22471 | 164312 | | JMP | E20-1,1 | NUMERIC |
| 0923 | 22472 | 066372 | | JMP | LENS1 | STRING |
| 0924* | | | | | | |
| 0925 | 22473 | 066465 | | JMP | INDS2 | |
| 0926* | | | | | | |
| 0927* | | | | | | |
| 0928 | 22474 | 062450 | EVAL | JSM | CKMOD | |
| 0929 | 22475 | 025400 | | LDB | BADDR | |
| 0930 | 22476 | 060563 | | JSM | SLWST | SAVE BADDR |
| 0931 | 22477 | 031645 | | STA | PTR1 | |
| 0932 | 22500 | 045654 | | ISZ | TNULL | NULL STRING? |
| 0933 | 22501 | 066503 | | JMP | *+2 | NO |
| 0934 | 22502 | 066534 | | JMP | VALER | YES |
| 0935 | 22503 | 021412 | | LDA | LSTPT | SET BUFFER |
| 0936 | 22504 | 070070 | | SIA | *+1 | START TO |
| 0937 | 22505 | 070744 | | SAL | 1 | LOW STACK |
| 0938 | 22506 | 031400 | | STA | BADDR | POINTER PLUS 1 |
| 0939 | 22507 | 031646 | | STA | PTR2 | SAVE BUFFER START |
| 0940 | 22510 | 021413 | | LDA | HSTPT | SET BUFFER |
| 0941 | 22511 | 070744 | | SAL | 1 | END TO HIGH |
| 0942 | 22512 | 031623 | | STA | PEND | STACK POINTER PLUS 1 |
| 0943 | 22513 | 027645 | EVAL1 | LDB | PTR1A | |
| 0944 | 22514 | 061041 | | JSM | GETCH | GET A SOURCE CHAR |
| 0945 | 22515 | 061016 | | JSM | ONEXT | PUT IN BUFFER |
| 0946 | 22516 | 021400 | | LDA | BADDR | |
| 0947 | 22517 | 011623 | | CPA | PEND | OVERFLOW? |
| 0948 | 22520 | 064561 | | JMP | MER8 | YES, ERROR |
| 0949 | 22521 | 045654 | | ISZ | TNULL | NO, DONE? |
| 0950 | 22522 | 066513 | | JMP | EVAL1 | NO |
| 0951 | 22523 | 020055 | | LDA | 1 | TERMINATE THE BUFFER |
| 0952 | 22524 | 061016 | | JSM | ONEXT | WITH A NON-DIGIT CHARACTER |
| 0953 | 22525 | 021646 | | LDA | PTR2 | |
| 0954 | 22526 | 031400 | | STA | BADDR | INITIALIZE BADDR |
| 0955 | 22527 | 061156 | | JSM | CONST | |
| 0956 | 22530 | 066534 | | JMP | VALER | |
| 0957 | 22531 | 060567 | | JSM | UNSTK | |
| 0958 | 22532 | 031400 | | STA | BADDR | RESTORE BADDR |
| 0959 | 22533 | 066365 | | JMP | ELEN2 | |
| 0960 | 22534 | 160205 | VALER | JSM | AERRA,1 | |
| 0961 | 22535 | 177664 | | DEC | -76 | |
| 0962* | | | | | | |
| 0963* | | | | | | |
| 0964* | | | | | | |
| 0965* | | | | | | |
| 0966 | 22536 | 121472 | GVAR | LDA | TEMPS,1 | GET LEFT PAREN COUNT |
| 0967 | 22537 | 031646 | | STA | PTR2 | AND SAVE |
| 0968 | 22540 | 021644 | | LDA | ST+4 | SAVE STRING |
| 0969 | 22541 | 031651 | | STA | TEMP6 | OCCURRED FLAG |
| 0970 | 22542 | 061055 | | JSM | GNEXT | GET NEXT CHAR |
| 0971 | 22543 | 055400 | | DSZ | BADDR | |
| 0972 | 22544 | 045706 | | ISZ | SBPTR | |
| 0973 | 22545 | 060521 | | JSM | LPCKE | DEMAND LEFT PAREN |
| 0974 | 22546 | 045400 | | ISZ | BADDR | |
| 0975 | 22547 | 066034 | | JMP | SVARO | |
| 0976* | | | | | | |

455

442

1,444,141

442

PAGE 339

STRING OPTION BLOCK

0979* STATEMENT EXECUTION

0980*

0981*

0982*

LET EXECUTION

0983*

0984*

0985*

0986 22550 063357 GPTRS JSM GADD

0987 22551 121472 LDA TEMPS,I

0988 22552 045472 ISZ TEMPS

0989 22553 031644 STA INTMP SAVE RELATIONAL OPERATOR

0990 22554 125472 LDB TEMPS,I

0991 22555 074442 SBR I0

0992 22556 014012 CPB .I QUOTE?

0993 22557 066572 JMP GPI YES

0994 22560 050065 AND OPMSK NO

0995 22561 010065 CPA OPMSK NON-FORMULA OPERATOR?

0996 22562 045472 ISZ TEMPS YES

0997 22563 055472 GP2 DSZ TEMPS

0998 22564 063357 JSM GADD GET SOURCE STRING ADDRESS

0999 22565 024133 LDB M2

1000 22566 063421 JSM PSTR EVALUATE SOURCE STRING

1001 22567 035647 XFER STB TMPRE SAVE LENGTH

1002 22570 031645 STA PTR1

1003 22571 170402 RET

1004 22572 063551 GPI JSM GQUOT GET QUOTE FIELD

1005 22573 066567 JMP XFER

1006* 22574 062550 LTEX JSM GPTRS GET DEST STRING POINTERS

1007 22575 024132 LT4 LDB M1 NO

1008 22576 063421 JSM PSTR EVALUATE DESTINATION STRING

1009 22577 074410 ISZ LT2+1 NULL STRING?

1010 22600 021416 LDA MODE

1011 22601 070250 SZA LT2 CALC MODE?

1012 22602 160170 JSM CLERA,I YES

1013 22603 021647 LDA TMPRE

1014 22604 025645 LDB PTR1

1015 22605 061330 JSM NPWR DISPLAY THE STRING

1016 22606 062610 LT2 JSM TRSTR

1017 22607 164172 JMP XEC4A,I

1018

1019*

1020*

1021*

STRING TRANSFER SUBROUTINE

1022*

1023*

1024 22610 021645 TRSTR LDA PTR1

1025 22611 070076 TCA

1026 22612 001742 ADA TMP2

1027 22613 070413 SAP TR1

1028 22614 045654 TR2 ISZ TNULL FORWARD TRANSFER?

1029 22615 066617 JMP *+2 YES

1030 22616 170402 RET

1031 22617 063255 JSM FSCH GET NEXT SOURCE CHAR

1032 22620 027731 LT5 LDB TMP2A

1033 22621 061017 JSM OUTCR STORE IT

1034 22622 066614 JMP TR2 LOOK FOR MORE

456

PAGE 340

STRING OPTION BLOCK

| | | | | | | |
|-------|-------|--------|-------|-----|---------|------------------------|
| 1035 | 22623 | 021654 | TR1 | LDA | TNULL | |
| 1036 | 22624 | 070070 | | SIA | *+1 | |
| 1037 | 22625 | 070056 | | CMA | | |
| 1038 | 22626 | 070137 | | LDB | A | |
| 1039 | 22627 | 005645 | | ADB | PTR1 | |
| 1040 | 22630 | 035645 | | STB | PTR1 | POINT TO END OF SOURCE |
| 1041 | 22631 | 001742 | | ADA | TMP2 | |
| 1042 | 22632 | 031742 | | STA | TMP2 | POINT TO END OF DEST |
| 1043 | 22633 | 021654 | | LDA | TNULL | |
| 1044 | 22634 | 070056 | | CMA | | |
| 1045 | 22635 | 025647 | | LDB | TMPRE | |
| 1046 | 22636 | 074076 | | TCB | | |
| 1047 | 22637 | 070037 | | ADB | A | |
| 1048 | 22640 | 074113 | | SBP | *+2 | TOO MUCH SOURCE? |
| 1049 | 22641 | 074742 | | SBR | 16 | YES, NO BLANKS NEEDED |
| 1050 | 22642 | 035647 | | STB | TMPRE | STORE BLANK COUNT |
| 1051 | 22643 | 045654 | TR4 | ISZ | TNULL | |
| 1052 | 22644 | 066646 | | JMP | *+2 | |
| 1053 | 22645 | 170402 | | RET | | |
| 1054 | 22646 | 025647 | | LDB | TMPRE | |
| 1055 | 22647 | 074250 | | SZB | TR3 | MORE BLANKS? |
| 1056 | 22650 | 020041 | | LDA | .32 | YES |
| 1057 | 22651 | 055647 | | DSZ | TMPRE | |
| 1058 | 22652 | 066653 | | JMP | *+1 | |
| 1059 | 22653 | 066657 | | JMP | TR5 | |
| 1060 | 22654 | 027645 | TR3 | LDB | PTR1A | |
| 1061 | 22655 | 061041 | | JSM | GETCH | |
| 1062 | 22656 | 055645 | | DSZ | PTR1 | |
| 1063 | 22657 | 027731 | TR5 | LDB | TMP2A | |
| 1064 | 22660 | 061017 | | JSM | OUTCR | |
| 1065 | 22661 | 055742 | | DSZ | TMP2 | |
| 1066 | 22662 | 055742 | | DSZ | TMP2 | |
| 1067 | 22663 | 055645 | | DSZ | PTR1 | |
| 1068 | 22664 | 066643 | | JMP | TR4 | NO |
| 1069* | | | | | | |
| 1070 | 22665 | 045654 | FINCH | ISZ | TNULL | MORE TRANSFER STRING |
| 1071 | 22666 | 066670 | | JMP | *+2 | YES |
| 1072 | 22667 | 170402 | | RET | | NO |
| 1073 | 22670 | 061040 | | JSM | GETCH-1 | |
| 1074 | 22671 | 010031 | | CPA | EOL | END OF LINE? |
| 1075 | 22672 | 170402 | | RET | | YES |
| 1076 | 22673 | 010043 | | CPA | B42 | YES, QUOTE? |
| 1077 | 22674 | 170402 | | RET | | YES |
| 1078 | 22675 | 027731 | | LDB | TMP2A | NO, |
| 1079 | 22676 | 061017 | | JSM | OUTCR | STORE IT |
| 1080 | 22677 | 066665 | | JMP | FINCH | |
| 1081* | | | | | | |

PAGE 341

STRING OPTION BLOCK

1083*

1084*

1085*

1086*

1087*

INPUT STATEMENT EXECUTION

| | | | | | | |
|------|-------|--------|-------|-----|---------|---|
| 1088 | 22700 | 063013 | EINPT | JSM | ESAVE | SAVE CURRENT PAGE ADDRESS |
| 1089 | 22701 | 020112 | | LDA | RSTKA | |
| 1090 | 22702 | 031777 | | STA | RSPTR | RESET STACK |
| 1091 | 22703 | 021622 | | LDA | TERM | |
| 1092 | 22704 | 072110 | | RZA | *+2 | |
| 1093 | 22705 | 160170 | | JSM | CLERA,I | |
| 1094 | 22706 | 021472 | | LDA | TEMPS | |
| 1095 | 22707 | 000133 | | ADA | M2 | |
| 1096 | 22710 | 031641 | | STA | SPRDD | SAVE START OF INPUT STATEMENT |
| 1097 | 22711 | 021641 | EIN2 | LDA | SPRDD | |
| 1098 | 22712 | 031421 | | STA | PRADD | INPUT STATEMENT |
| 1099 | 22713 | 021472 | | LDA | TEMPS | |
| 1100 | 22714 | 031640 | | STA | TEPPP | |
| 1101 | 22715 | 121472 | | LDA | TEMPS,I | |
| 1102 | 22716 | 050040 | | AND | B37 | |
| 1103 | 22717 | 010076 | | CPA | B36 | STRING VARIABLE? |
| 1104 | 22720 | 067017 | | JMP | INSTR | YES |
| 1105 | 22721 | 023010 | | LDA | INLNK | |
| 1106 | 22722 | 001643 | | ADA | SAVE | |
| 1107 | 22723 | 031432 | | STA | DRQST | SET UP RETURN LINK |
| 1108 | 22724 | 164173 | INRET | JMP | PEXMA,I | CALL FOR INPUT |
| 1109 | 22725 | 055400 | | DSZ | BADDR | |
| 1110 | 22726 | 025641 | | LDB | SPRDD | |
| 1111 | 22727 | 061007 | | JSM | NEXTA | SET PRADD BACK TO END |
| 1112 | 22730 | 035421 | | STB | PRADD | OF INPUT STATEMENT |
| 1113 | 22731 | 021472 | | LDA | TEMPS | |
| 1114 | 22732 | 031640 | | STA | TEPPP | SAVE START OF VARIABLE |
| 1115 | 22733 | 160201 | EIN1 | JSM | FORMA,I | GET THE ADDRESS OF THE VARIABLE
IN HS |
| 1116 | 22734 | 024017 | | LDB | 2 | |
| 1117 | 22735 | 035706 | | STB | SBPTR | AIM SBPTR AT ROM |
| 1118 | 22736 | 027007 | | LDB | INLN1 | |
| 1119 | 22737 | 005643 | | ADB | SAVE | |
| 1120 | 22740 | 035432 | | STB | DRQST | |
| 1121 | 22741 | 061156 | | JSM | CONST | BUILD A NUMBER IN AR2 |
| 1122 | 22742 | 066765 | | JMP | INCH5 | P+1 — NO NUMBER FOUND, SO INC
HSTPTP |
| 1123 | 22743 | 070742 | | SAR | 16 | |
| 1124 | 22744 | 031432 | | STA | DRQST | CLEAR FLAG |
| 1125 | 22745 | 060412 | | JSM | RESULT | SAVE ARE AT HSTPT,I |
| 1126 | 22746 | 025472 | | LDB | TEMPS | |
| 1127 | 22747 | 015421 | | CPB | PRADD | END OF STATEMENT |
| 1128 | 22750 | 067011 | | JMP | CXEC4 | YES |
| 1129 | 22751 | 035640 | EIN3 | STB | TEPPP | RESET THE TEMPORARY POINTER |
| 1130 | 22752 | 055400 | | DSZ | BADDR | NO — DINC BADDR TO LOOK AT THE
LAST CH |
| 1131 | 22753 | 061066 | | JSM | GETSK | GET LAST CHAR. |
| 1132 | 22754 | 067005 | | JMP | INCH1 | |
| 1133 | 22755 | 013642 | | CPA | B54 | COMMA? |
| 1134 | 22756 | 066760 | | JMP | *+2 | YES |
| 1135 | 22757 | 066777 | | JMP | INCH2 | NO |
| 1136 | 22760 | 121472 | | LDA | TEMPS,I | |
| 1137 | 22761 | 050040 | | AND | B37 | |
| 1138 | 22762 | 010076 | | CPA | B36 | STRING VARIABLE? |

PAGE 342

STRING OPTION BLOCK

| | | | | | | |
|-------|-------|--------|-------|-----|-----------|--------------------------|
| 1139 | 22763 | 067024 | | JMP | EIN4 | YES |
| 1140 | 22764 | 066733 | | JMP | EIN1 | NO |
| 1141* | | | | | | |
| 1142 | 22765 | 160170 | INCH5 | JSM | CLERA,I | |
| 1143 | 22766 | 023643 | | LDA | .17 | |
| 1144 | 22767 | 027712 | | LDB | ER3A | |
| 1145 | 22770 | 063137 | | JSM | ER4+2 | |
| 1146 | 22771 | 063135 | | JSM | ER4 | |
| 1147 | 22772 | 021640 | INCHT | LDA | TEPPP | |
| 1148 | 22773 | 031472 | | STA | TEMPS | RESTORE VARIABLE POINTER |
| 1149 | 22774 | 045413 | | ISZ | HSTPT | |
| 1150 | 22775 | 045413 | | ISZ | HSTPT | |
| 1151 | 22776 | 066711 | | JMP | EIN2 | |
| 1152* | | | | | | |
| 1153 | 22777 | 160170 | INCH2 | JSM | CLERA,I | |
| 1154 | 23000 | 020034 | | LDA | .21 | |
| 1155 | 23001 | 027725 | | LDB | ER1A | |
| 1156 | 23002 | 063137 | | JSM | ER4+2 | |
| 1157 | 23003 | 063135 | | JSM | ER4 | |
| 1158 | 23004 | 066711 | | JMP | EIN2 | |
| 1159* | | | | | | |
| 1160 | 23005 | 160170 | INCH1 | JSM | CLERA,I | |
| 1161 | 23006 | 066711 | | JMP | EIN2 | |
| 1162 | 23007 | 000772 | INLN1 | ABS | INCHT-ID1 | |
| 1163 | 23010 | 000724 | INLNK | ABS | INRET-ID1 | |
| 1164* | | | | | | |
| 1165 | 23011 | 160170 | CXEC4 | JSM | CLERA,I | |
| 1166 | 23012 | 164172 | | JMP | XEC4A,I | |
| 1167* | | | | | | |
| 1168* | | | | | | |
| 1169* | | | | | | |
| 1170 | 23013 | 074117 | ESAVE | LDA | B | |
| 1171 | 23014 | 050160 | | AND | B174K | |
| 1172 | 23015 | 031643 | | STA | SAVE | |
| 1173 | 23016 | 170402 | | RET | | |
| 1174* | | | | | | |
| 1175* | | | | | | |
| 1176 | 23017 | 070742 | INSTR | SAR | 16 | |
| 1177 | 23020 | 031701 | | STA | NDISP | |
| 1178 | 23021 | 031702 | | STA | NPRIN | |
| 1179 | 23022 | 031707 | | STA | BLANK | DON'T IGNORE BLANKS |
| 1180 | 23023 | 063142 | | JSM | INCAL | |
| 1181 | 23024 | 063357 | EIN4 | JSM | GADD | GET STRING ADDRESS |
| 1182 | 23025 | 025412 | | LDB | LSTPT | EXTRACT |
| 1183 | 23026 | 004133 | | ADB | M2 | PHYSICAL |
| 1184 | 23027 | 074517 | | LDA | B,I | LENGTH OF |
| 1185 | 23030 | 070002 | | SAR | 1 | |
| 1186 | 23031 | 000132 | | ADA | M1 | DESTINATION |
| 1187 | 23032 | 070517 | | LDA | A,I | STRING |
| 1188 | 23033 | 070342 | | SAR | 8 | |
| 1189 | 23034 | 070076 | | TCA | | |
| 1190 | 23035 | 004012 | | ADB | .1 | SUBTRACT FIRST |
| 1191 | 23036 | 074417 | | ADA | B,I | SUBSCRIPT |
| 1192 | 23037 | 000132 | | ADA | M1 | |
| 1193 | 23040 | 070076 | | TCA | | STORE IT AS END OF |
| 1194 | 23041 | 031647 | | STA | TMPRE | UNSPECIFIED DEST STRING |

PAGE 343

STRING OPTION BLOCK

| | | | | | |
|------|-------|--------|-----|---------|---------------------------------|
| 1195 | 23042 | 024132 | LDB | M1 | PREPARE |
| 1196 | 23043 | 063421 | JSM | PSTR | DESTINATION STRING |
| 1197 | 23044 | 021742 | LDA | TMP2 | |
| 1198 | 23045 | 031645 | STA | PTR1 | |
| 1199 | 23046 | 025654 | LDB | TNULL | SAVE LENGTH |
| 1200 | 23047 | 035644 | STB | INTMP | ALLOWANCE |
| 1201 | 23050 | 061055 | JSM | GNEXT | |
| 1202 | 23051 | 067055 | JMP | EINS | |
| 1203 | 23052 | 063142 | JSM | INCAL | GET INPUT RECORD |
| 1204 | 23053 | 061062 | JSM | GFRST | |
| 1205 | 23054 | 067054 | JMP | * | CHECKSUM WORD |
| 1206 | 23055 | 025645 | LDB | PTR1 | |
| 1207 | 23056 | 035742 | STB | TMP2 | RESTORE STRING POINTER |
| 1208 | 23057 | 010043 | CPA | B42 | QUOTE? |
| 1209 | 23060 | 067062 | JMP | *+2 | YES |
| 1210 | 23061 | 055400 | DSZ | BADDR | NO |
| 1211 | 23062 | 062665 | JSM | FINCH | TRANSFER THE STRING |
| 1212 | 23063 | 074742 | SBR | 16 | ALL REQUESTED |
| 1213 | 23064 | 015654 | CPB | TNULL | CHARACTERS TRANSFERRED? |
| 1214 | 23065 | 067112 | JMP | EIN21 | YES |
| 1215 | 23066 | 015656 | CPB | PS1 | NO, TRANSFER LENGTH SPECIFIED |
| 1216 | 23067 | 067106 | JMP | EIN20 | NO |
| 1217 | 23070 | 031644 | STA | INTMP | SAVE EOR CHAR |
| 1218 | 23071 | 070742 | SAR | 16 | |
| 1219 | 23072 | 031647 | STA | TMPRE | |
| 1220 | 23073 | 001654 | ADA | TNULL | TRANSFER |
| 1221 | 23074 | 000132 | ADA | M1 | |
| 1222 | 23075 | 031654 | STA | TNULL | |
| 1223 | 23076 | 062614 | JSM | TR2 | |
| 1224 | 23077 | 010031 | CPA | EOL | TRANSFER ENDED BY EOL? |
| 1225 | 23100 | 067102 | JMP | EINP6 | YES |
| 1226 | 23101 | 061055 | JSM | GNEXT | NO, WAS A" |
| 1227 | 23102 | 025641 | LDB | SPRDD | |
| 1228 | 23103 | 061007 | JSM | NEXTA | SET PRADD BACK TO |
| 1229 | 23104 | 035421 | STB | PRADD | END OF INPUT STATEMENT |
| 1230 | 23105 | 066746 | JMP | EIN3-3 | LOOK FOR END |
| 1231 | 23106 | 125651 | LDB | TEMP6,I | SET LOGICAL |
| 1232 | 23107 | 005654 | ADB | TNULL | TO ACTUAL |
| 1233 | 23110 | 135651 | STB | TEMPS,I | STRING LENGTH |
| 1234 | 23111 | 067077 | JMP | EIN18 | |
| 1235 | 23112 | 015656 | CPB | PS1 | LENGTH OF TRANSFER SPECIFIED? |
| 1236 | 23113 | 067121 | JMP | EIN15 | NO |
| 1237 | 23114 | 061066 | JSM | GETSK | YES, END OF LINE? |
| 1238 | 23115 | 067102 | JMP | EINP6 | YES |
| 1239 | 23116 | 010043 | CPA | B42 | |
| 1240 | 23117 | 067101 | JMP | EIN19 | YES |
| 1241 | 23120 | 067114 | JMP | EIN16 | NO, LOOK FOR "OR EOL |
| 1242 | 23121 | 061066 | JSM | GETSK | EOL? |
| 1243 | 23122 | 067102 | JMP | EINP6 | YES |
| 1244 | 23123 | 010043 | CPA | B42 | NO, CLOSING QUOTE? |
| 1245 | 23124 | 067101 | JMP | EIN19 | YES |
| 1246 | 23125 | 021644 | LDA | INTMP | NO, DESTINATION STRING EXCEEDED |
| 1247 | 23126 | 031654 | STA | TNULL | RESTORE DESTINATION |
| 1248 | 23127 | 160170 | JSM | CLERA,I | |
| 1249 | 23130 | 020033 | LDA | 16 | |
| 1250 | 23131 | 027726 | LDB | ER2A | |

PAGE 344

STRING OPTION BLOCK

| | | | | | | |
|-------|-------|--------|-------|-----|---------|-----------------------------|
| 1251 | 23132 | 063137 | | JSM | ER4+2 | |
| 1252 | 23133 | 063135 | | JSM | ER4 | |
| 1253 | 23134 | 067052 | | JMP | EIN14 | |
| 1254* | | | | | | |
| 1255* | | | | | | |
| 1256* | | | | | | |
| 1257 | 23135 | 020025 | ER4 | LDA | .8 | |
| 1258 | 23136 | 027724 | | LDB | ER4A | |
| 1259 | 23137 | 005643 | | ADB | SAVE | |
| 1260 | 23140 | 005643 | | ADB | SAVE | |
| 1261 | 23141 | 065330 | | JMP | NPWR | |
| 1262* | | | | | | |
| 1263 | 23142 | 020012 | INCAL | LDA | .1 | |
| 1264 | 23143 | 031432 | | STA | DRQST | |
| 1265 | 23144 | 027727 | | LDB | QMRKA | |
| 1266 | 23145 | 061330 | | JSM | NPWR | OUTPUT A? |
| 1267 | 23146 | 161462 | | JSM | REEDL,I | GET INPUT RECORD |
| 1268 | 23147 | 067164 | | JMP | INC1 | NULL RECORD |
| 1269 | 23150 | 061062 | | JSM | GFRST | |
| 1270 | 23151 | 067164 | | JMP | INC1 | YES |
| 1271 | 23152 | 021432 | | LDA | DRQST | |
| 1272 | 23153 | 072210 | | RZA | *+4 | EXECUTION EXIT OCCURRED? |
| 1273 | 23154 | 020041 | | LDA | .32 | YES |
| 1274 | 23155 | 031707 | | STA | BLANK | |
| 1275 | 23156 | 167613 | | JMP | REEDY,I | |
| 1276 | 23157 | 074742 | | SBR | 16 | |
| 1277 | 23160 | 035432 | | STB | DRQST | |
| 1278 | 23161 | 035416 | | STB | MODE | |
| 1279 | 23162 | 055400 | | DSZ | BADDR | |
| 1280 | 23163 | 170402 | | RET | | |
| 1281 | 23164 | 021432 | INC1 | LDA | DRQST | |
| 1282 | 23165 | 070150 | | SZA | INC2 | EXECUTION EXIT OCCURRED? |
| 1283 | 23166 | 160170 | | JSM | CLERA,I | NO |
| 1284 | 23167 | 067142 | | JMP | INCAL | |
| 1285 | 23170 | 167612 | INC2 | JMP | REEDX,I | |
| 1286* | | | | | | |
| 1287* | | | | | | |
| 1288* | | | | | | |
| 1289* | | | | | | |
| 1290* | | | | | | |
| 1291* | | | | | | |
| 1292* | | | | | | |
| 1293 | 23171 | 062550 | IFEX | JSM | GPTRS | GET STRING PARAMETERS |
| 1294 | 23172 | 074056 | | CMB | | |
| 1295 | 23173 | 035640 | | STB | CP0 | |
| 1296 | 23174 | 020133 | | LDA | M2 | LENGTH |
| 1297 | 23175 | 063421 | | JSM | PSTR | PREPARE TEST STRING |
| 1298 | 23176 | 035641 | | STB | CP1 | SAVE ACTUAL LENGTH |
| 1299 | 23177 | 045640 | COMP2 | ISZ | CP0 | MORE SPECIFIED STRING? |
| 1300 | 23200 | 067203 | | JMP | COMP3 | YES |
| 1301 | 23201 | 074742 | | SBR | 16 | NO, LOAD A |
| 1302 | 23202 | 067206 | | JMP | COMP4 | NULL CHARACTER |
| 1303 | 23203 | 027645 | COMP3 | LDB | PTRIA | |
| 1304 | 23204 | 061041 | | JSM | GETCH | |
| 1305 | 23205 | 071037 | | LDB | A | |
| 1306 | 23206 | 045654 | COMP4 | ISZ | TNULL | MORE SPECIFIED TEST STRING? |

PAGE 345

STRING OPTION BLOCK

| | | | | | | |
|-------|-------|--------|-------|-----|---------|--------------------------|
| 1307 | 23207 | 067216 | | JMP | COMP6 | YES |
| 1308 | 23210 | 070742 | | SAR | 16 | NO |
| 1309 | 23211 | 074076 | COMP5 | TCB | | COMPARE |
| 1310 | 23212 | 074017 | | ADA | B | CHARACTERS |
| 1311 | 23213 | 072110 | | RZA | *+2 | EXIT ON NOT EQUAL |
| 1312 | 23214 | 077150 | | RZB | COMP2 | OR BOTH NULL CHARACTERS |
| 1313 | 23215 | 067231 | | JMP | BRANC | EXIT TO BRANCH |
| 1314 | 23216 | 021641 | COMP6 | LDA | CP1 | MORE ACTUAL |
| 1315 | 23217 | 072150 | | RZA | COMP7 | TEST STRING? |
| 1316 | 23220 | 020041 | | LDA | .32 | NO |
| 1317 | 23221 | 067211 | | JMP | COMP5 | |
| 1318 | 23222 | 000132 | COMP7 | ADA | M1 | |
| 1319 | 23223 | 031641 | | STA | CP1 | |
| 1320 | 23224 | 035650 | | STB | ST+8 | |
| 1321 | 23225 | 027731 | | LDB | TMP2A | EXTRACT NEXT |
| 1322 | 23226 | 061041 | | JSM | GETCH | TEST |
| 1323 | 23227 | 025650 | | LDB | ST+8 | |
| 1324 | 23230 | 067211 | | JMP | COMP5 | CHARACTER |
| 1325* | | | | | | |
| 1326 | 23231 | 070137 | BRANC | LDB | A | |
| 1327 | 23232 | 021664 | | LDA | INTMP | |
| 1328 | 23233 | 050070 | | AND | OPDMK | |
| 1329 | 23234 | 070202 | | SAR | 5 | |
| 1330 | 23235 | 074153 | | SBP | BR1 | LESS THAN? |
| 1331 | 23236 | 070211 | | SLA | GOTOE | YES, EVEN OP CODE? |
| 1332 | 23237 | 164172 | | JMP | XEC4A,I | NO |
| 1333 | 23240 | 076310 | BR1 | RZB | BR2 | EQUAL? |
| 1334 | 23241 | 010022 | | CPA | .5 | YES, - OP? |
| 1335 | 23242 | 167577 | GOTOE | JMP | EGOS1,I | YES |
| 1336 | 23243 | 000135 | | ADA | M4 | |
| 1337 | 23244 | 071712 | | SAM | *-2 | |
| 1338 | 23245 | 164172 | | JMP | XEC4A,I | |
| 1339 | 23246 | 010023 | BR2 | CPA | .6 | <? |
| 1340 | 23247 | 164172 | | JMP | XEC4A,I | |
| 1341 | 23250 | 010022 | | CPA | .5 | =? |
| 1342 | 23251 | 164172 | | JMP | XEC4A,I | |
| 1343 | 23252 | 010017 | | CPA | .2 | |
| 1344 | 23253 | 164172 | | JMP | XEC4A,I | |
| 1345 | 23254 | 167577 | | JMP | EGOS1,I | |
| 1346* | | | | | | |
| 1347* | | | | | | |
| 1348 | 23255 | 025647 | FSCH | LDB | TMPRE | |
| 1349 | 23256 | 076150 | | RZB | *+3 | OUT OF STRING? |
| 1350 | 23257 | 020041 | | LDA | .32 | YES, GET A |
| 1351 | 23260 | 170402 | | RET | | BLANK |
| 1352 | 23261 | 004132 | | ADB | M1 | |
| 1353 | 23262 | 035647 | | STB | TMPRE | |
| 1354 | 23263 | 027645 | | LDB | PTR1A | |
| 1355 | 23264 | 065041 | | JMP | GETCH | |
| 1356 | | | | LST | | |
| 1357* | | | | | | |
| 1358 | 23265 | 121472 | EREAD | LDA | TEMPS,I | |
| 1359 | 23266 | 050040 | | AND | B37 | |
| 1360 | 23267 | 010076 | | CPA | B36 | STRING VARIABLE? |
| 1361 | 23270 | 067310 | | JMP | RDSTR | YES |
| 1362 | 23271 | 160201 | | JSM | FORMA,I | NO, GET VARIABLE ADDRESS |

PAGE 346

STRING OPTION BLOCK

| | | | | | | |
|-------|-------|--------|-------|-----|---------|--------------------------------|
| 1367 | 23272 | 163611 | ER1 | JSM | SWITH,I | EXCHANGE TEMPS AND NXTDT |
| 1364 | 23273 | 015414 | | CPB | DSTRT | NEED NEW DATA STATEMENT? |
| 1365 | 23274 | 063334 | | JSM | .FDAI | YES |
| 1366 | 23275 | 163661 | | JSM | SWITH,I | NO |
| 1367 | 23276 | 125415 | | LDB | NXTDT,I | |
| 1368 | 23277 | 074152 | | SBM | *+3 | NUMERIC DATA IS NEXT? |
| 1369 | 23300 | 160205 | | JSM | AERRA,I | NO |
| 1370 | 23301 | 177665 | | DEC | -75 | |
| 1371 | 23302 | 160256 | | JSM | FDAAA,I | YES |
| 1372 | 23303 | 060412 | | JSM | RESULT | |
| 1373 | 23304 | 021421 | ER2 | LDA | PRADD | |
| 1374 | 23305 | 011472 | | CPA | TEMPS | MORE VARIABLES? |
| 1375 | 23306 | 164172 | | JMP | XEC4A,I | |
| 1376 | 23307 | 067265 | | JMP | EREAD | |
| 1377 | 23310 | 063357 | RDSTR | JSM | GADD | GET STRING VARIABLE PARAMETERS |
| 1378 | 23311 | 163611 | | JSM | SWITH,I | |
| 1379 | 23312 | 025472 | ER3 | LDB | TEMPS | |
| 1380 | 23313 | 015414 | | CPB | DSTRT | NEED NEW DATA STMT? |
| 1381 | 23314 | 063334 | | JSM | .FDAI | YES |
| 1382 | 23315 | 045472 | | ISZ | TEMPS | |
| 1383 | 23316 | 125472 | | LDB | TEMPS,I | |
| 1384 | 23317 | 074442 | | SBR | 10 | |
| 1385 | 23320 | 014012 | | CPB | .1 | QUOTE? |
| 1386 | 23321 | 067234 | | JMP | *+3 | YES |
| 1387 | 23322 | 160205 | | JSM | AERRA,I | YES |
| 1388 | 23323 | 177665 | | DEC | -75 | |
| 1389 | 23324 | 063351 | | JSM | GQUOT | NO |
| 1390 | 23325 | 035647 | | STB | TMPRE | |
| 1391 | 23326 | 031645 | | STA | PTR1 | |
| 1392 | 23327 | 024132 | | LDB | M1 | PREPARE |
| 1393 | 23330 | 063421 | | JSM | PSTR | DESTINATION STRING |
| 1394 | 23331 | 062610 | | JSM | TRSTR | TRANSFER THE STRING |
| 1395 | 23332 | 163611 | | JSM | SWITH,I | |
| 1396 | 23333 | 067304 | | JMP | ER2 | |
| 1397* | | | | | | |
| 1398* | | | | | | |
| 1399 | 23334 | 021421 | .FDAI | LDA | PRADD | |
| 1400 | 23335 | 031644 | | STA | INTMP | |
| 1401 | 23336 | 035421 | | STB | PRADD | |
| 1402 | 23337 | 025421 | FD2 | LDB | PRADD | |
| 1403 | 23340 | 015423 | | CPB | PBPTR | END OF PROGRAM? |
| 1404 | 23341 | 164350 | | JMP | E50-1,I | OUT OF DATA |
| 1405 | 23342 | 035472 | | STB | TEMPS | |
| 1406 | 23343 | 060577 | | JSM | NEXTL+3 | |
| 1407 | 23344 | 001466 | | ADA | LOCL1 | ENCODE PAGE AND OP CODE |
| 1408 | 23345 | 013732 | | CPA | MSDAT | MAIN SYSTEM DATA STMT? |
| 1409 | 23346 | 067352 | | JMP | FD3 | YES |
| 1410 | 23347 | 013733 | | CPA | STDAT | NO, STRING DATA STMT? |
| 1411 | 23350 | 067352 | | JMP | FD3 | YES |
| 1412 | 23351 | 067337 | | JMP | FD2 | NO, KEEP LOOKING |
| 1413 | 23352 | 025421 | FD3 | LDB | PRADD | |
| 1414 | 23353 | 035414 | | STB | DSTRT | |
| 1415 | 23354 | 025644 | | LDB | INTMP | |
| 1416 | 23355 | 035421 | | STB | PRADD | |
| 1417 | 23356 | 170402 | | RET | | |
| 1418* | | | | | | |

PAGE 347

STRING OPTION BLOCK

```

1422*
1423*   GADD SUBROUTINE
1424*
1425*   FETCH STRING ADDRESS FROM SYMBOL TABLE
1426*
1427*   ENTRY CONDITIONS:
1428*
1429*       A=OPERAND
1430*       TEMPS POINTS TO STRING NAME
1431*
1432*   EXIT CONDITIONS:
1433*
1434*       STRING ADDRESS AND SUBSCRIPTS ARE
1435*       ON LOW STACK
1436*
1437   23357 060771 GADD   JSM  GTOPP   GET OPERAND
1438   23360 063567      JSM  GSYMB   GET SYMBOL ADDRESS
1439   23361 004132      ADB  M1
1440   23362 074537      LDB  B,I
1441   23363 074744      SBL  I
1442   23364 060563      JSM  SLWST   PUT ON STACK
1443   23365 045472      ISZ  TEMPS
1444   23366 021472      LDA  TEMPS
1445   23367 011421      CPA  PRADD   OUT OF CODE?
1446   23370 067375      JMP  GA2
1447   23371 121472      LDA  TEMPS,I
1448   23372 050065      AND  OPMSK   GET OP CODE
1449   23373 010125      CPA  LBOP   SUBSCRIPTED?
1450   23374 067401      JMP  GA1     YES
1451   23375 074742      GA2        SBR  16     NO
1452   23376 060563      JSM  SLWST
1453   23377 024132      LDB  M1
1454   23400 064563      JMP  SLWST
1455   23401 063412      GA1        JSM  SUBEV   EVALUATE SUBSCRIPTS
1456   23402 055472      DSZ  TEMPS
1457   23403 121472      LDA  TEMPS,I
1458   23404 070442      SAR  10
1459   23405 050040      AND  B37
1460   23406 010017      CPA  2      COMMA?
1461   23407 067412      JMP  SUBEV   YES
1462   23410 045472      ISZ  TEMPS
1463   23411 067377      JMP  GA1-2   NO
1464*
1465*   SUBEV SUBROUTINE
1466*
1467*   EVALUATE AN EXPRESSION, CONVERT TO
1468*   AN INTEGER, STORE ON LOW STACK
1469*
1470   23412 061217 SUBEV   JSM  FETCH   EVALUATE EXPRESSION
1471   23413 160225      JSM  FLTRA,I   CONVERT TO INTEGER
1472   23414 164341      JMP  E40+2,I
1473   23415 004132      ADB  M1
1474   23416 075712      SBM  *-2     0 OR NEGATIVE?
1475   23417 045472      ISZ  TEMPS
1476   23420 064563      JMP  SLWST   NO
1477*

```

PAGE 348

STRING OPTION BLOCK

| | | | | | |
|-------|-------|--------|---|-------------|--------------------------|
| 1479* | | | | | |
| 1480* | | PSTR | SUBROUTINE | | |
| 1481* | | | | | |
| 1482* | | | EVALUATE A STRING OPERAND | | |
| 1483* | | | | | |
| 1484* | | | ENTRY CONDITIONS: | | |
| 1485* | | | | | |
| 1486* | | | B=-2 FOR SOURCE STRINGS | | |
| 1487* | | | B=-1 FOR DESTINATION STRING | | |
| 1488* | | | LOW STACK HAS SECOND SUBSCRIPT. | | |
| 1489* | | | FIRST SUBSCRIPT, STRING ADDRESS. | | |
| 1490* | | | RESPECTIVELY. | | |
| 1491* | | | TMPRE=SOURCE LENGTH | | |
| 1492* | | | | | |
| 1493* | | | EXIT CONDITIONS: | | |
| 1494* | | | | | |
| 1495* | | | TMP2=STRING POINTER | | |
| 1496* | | | B=LENGTH | | |
| 1497* | | | TEMP6=STRING LENGTH WORD POINTER | | |
| 1498* | | | A=TMP2 IF SOURCE STRING | | |
| 1499* | | | TNULL=1, S COMPLEMENT OF REQUESTED LENGTH | | |
| 1500* | | | | | |
| 1501* | | | | | |
| 1502 | 23421 | 035655 | PSTR | STB PS0 | SAVE MODE FLAG |
| 1503 | 23422 | 025412 | | LDB LSTPT | |
| 1504 | 23423 | 004134 | | ADB M3 | |
| 1505 | 23424 | 035412 | | STB LSTPT | BUMP LOW STACK POINTER |
| 1506 | 23425 | 004012 | | ADB .1 | |
| 1507 | 23426 | 035656 | | STB PS1 | SET FLAG POSITIVE |
| 1508 | 23427 | 074517 | | LDA B,I | GET STRING ADDRESS |
| 1509 | 23430 | 031742 | | STA TMP2 | SAVE IT |
| 1510 | 23431 | 070002 | | SAR 1 | |
| 1511 | 23432 | 000132 | | ADA M1 | GET LENGTH WORD POINTER |
| 1512 | 23433 | 031651 | | STA TEMP6 | SAVE IT |
| 1513 | 23434 | 004012 | | ADB .1 | |
| 1514 | 23435 | 074517 | | LDA B,I | GET FIRST SUBSCRIPT |
| 1515 | 23436 | 031652 | | STA MPT | SAVE IT |
| 1516 | 23437 | 001742 | | ADA TMP2 | RECORD STRING |
| 1517 | 23440 | 031742 | | STA TMP2 | START POINTER |
| 1518 | 23441 | 004012 | PS2 | ADB .1 | |
| 1519 | 23442 | 074517 | | LDA B,I | GET SECOND SUBSCRIPT |
| 1520 | 23443 | 070513 | | SAP PSTR2 | SPECIFIED? |
| 1521 | 23444 | 011655 | | CPA PS0 | NO, SOURCE MODE? |
| 1522 | 23445 | 067451 | | JMP PSTR1 | NO |
| 1523 | 23446 | 121651 | | LDA TEMP6,I | YES |
| 1524 | 23447 | 050105 | | AND B377 | GET ACTUAL STRING LENGTH |
| 1525 | 23450 | 067454 | | JMP PSTR2-1 | |
| 1526 | 23451 | 031656 | PSTR1 | STA PS1 | SET FLAG TO -1 |
| 1527 | 23452 | 021647 | | LDA TMPRE | COMPUTE END OF |
| 1528 | 23453 | 001652 | | ADA MPT | STRING DESIGNATOR |
| 1529 | 23454 | 000132 | | ADA M1 | |
| 1530 | 23455 | 031653 | PSTR2 | STA NQT | SAVE IT |
| 1531 | 23456 | 070056 | | CMA | |
| 1532 | 23457 | 001652 | | ADA MPT | |
| 1533 | 23460 | 000132 | | ADA M1 | |
| 1534 | 23461 | 070152 | | SAM PSTR8 | NEGATIVE LENGTH? |

PAGE 349

STRING OPTION BLOCK

| | | | | | |
|-------|-------|--------|-----|---------|-------------------------------|
| 1535 | 23462 | 160205 | JSM | AERRA,I | YES |
| 1536 | 23463 | 177670 | DEC | -72 | |
| 1537 | 23464 | 031654 | STA | TNULL | SAVE 1,S COMPLEMENT OF LENGTH |
| 1538 | 23465 | 121651 | LDA | TEMP6,I | DOES |
| 1539 | 23466 | 050105 | AND | B377 | START OF |
| 1540 | 23467 | 070056 | CMA | | CHARACTER |
| 1541 | 23470 | 045655 | ISZ | PS0 | RELATE TO |
| 1542 | 23471 | 070070 | SIA | *+1 | PREVIOUS STRING |
| 1543 | 23472 | 001652 | ADA | MPT | VALUE |
| 1544 | 23473 | 070112 | SAM | *+2 | |
| 1545 | 23474 | 067515 | JMP | PSTR3 | |
| 1546 | 23475 | 121651 | LDA | TEMP6,I | YES |
| 1547 | 23476 | 045655 | ISZ | PS0 | DESTINATION MODE? |
| 1548 | 23477 | 070346 | RAR | 8 | YES, USE PHYSICAL LENGTH |
| 1549 | 23500 | 050105 | AND | B377 | NO USE ACTUAL LENGTH |
| 1550 | 23501 | 000012 | ADA | .1 | |
| 1551 | 23502 | 070076 | TCA | | |
| 1552 | 23503 | 001653 | ADA | NQT | |
| 1553 | 23504 | 024012 | LDB | .1 | |
| 1554 | 23505 | 015655 | CPB | PS0 | SOURCE MODE? |
| 1555 | 23506 | 067525 | JMP | PSTR5 | NO |
| 1556 | 23507 | 025654 | LDB | TNULL | SPECIFIED SOURCE |
| 1557 | 23510 | 000012 | ADA | .1 | STRING CONTAINED |
| 1558 | 23511 | 070452 | SAM | PSTR4 | WITHIN DEFINED SOURCE |
| | | | | | STRING? |
| 1559 | 23512 | 004012 | ADB | .1 | NO, |
| 1560 | 23513 | 070037 | ADB | A | CORRECT LENGTH |
| 1561 | 23514 | 067522 | JMP | PSTR4 | |
| 1562 | 23515 | 021655 | LDA | PS0 | SOURCE MODE? |
| 1563 | 23516 | 072150 | RZA | PSTR4-1 | YES |
| 1564 | 23517 | 160205 | JSM | AERRA,I | NO |
| 1565 | 23520 | 177667 | DEC | -73 | |
| 1566 | 23521 | 024132 | LDB | M1 | |
| 1567 | 23522 | 021742 | LDA | TMP2 | NO, GET STRING POINTER |
| 1568 | 23523 | 074056 | CMB | | |
| 1569 | 23524 | 170402 | RET | | |
| 1570 | 23525 | 070152 | SAM | *+3 | OVERFLOW? |
| 1571 | 23526 | 160205 | JSM | AERRA,I | YES |
| 1572 | 23527 | 177666 | DEC | -74 | |
| 1573 | 23530 | 045656 | ISZ | PS1 | NO, STRING END SPECIFIED? |
| 1574 | 23531 | 067543 | JMP | PSTR7 | YES |
| 1575 | 23532 | 045653 | ISZ | NQT | |
| 1576 | 23533 | 067534 | JMP | *+1 | |
| 1577 | 23534 | 121651 | LDA | TEMP6,I | NO |
| 1578 | 23535 | 050157 | AND | M256 | RESET |
| 1579 | 23536 | 041653 | IOR | NQT | STRING |
| 1580 | 23537 | 131651 | STA | TEMP6,I | ACTUAL LENGTH |
| 1581 | 23540 | 050105 | AND | B377 | |
| 1582 | 23541 | 070137 | LDB | A | |
| 1583 | 23542 | 170402 | RET | | |
| 1584 | 23543 | 121651 | LDA | TEMP6,I | IS NEW |
| 1585 | 23544 | 050105 | AND | B377 | DESTINATION |
| 1586 | 23545 | 070076 | TCA | | STRING |
| 1587 | 23546 | 001653 | ADA | NQT | LONGER |
| 1588 | 23547 | 071153 | SAP | PSTR6-2 | THAN OLD? |
| 1589 | 23550 | 170402 | RET | | NO |
| 1590* | | | | | |

PAGE 350

STRING OPTION BLOCK

1591*
 1592 23551 121472 GQUOT LDA TEMPS,I
 1593 23552 050105 AND B377 GET STRING LENGTH
 1594 23553 070137 LDB A
 1595 23554 021472 LDA TEMPS
 1596 23555 000012 ADA .1
 1597 23556 031651 STA TEMP6
 1598 23557 074117 LDA B
 1599 23560 070070 SIA *+1
 1600 23561 070002 SAR 1
 1601 23562 001651 ADA TEMP6
 1602 23563 031472 STA TEMPS UPDATE TEMPS
 1603 23564 021651 LDA TEMP6
 1604 23565 070744 SAL 1
 1605 23566 170402 RET

FIND VARIABLE ADDRESS

1611 23567 160177 GSYMB JSM SSYMA,I SYMBOL IN TABLE?
 1612 23570 160340 JSM E40+1,I NO
 1613 23571 170402 RET
 1614*
 1615*
 1616*
 1617 23572 164172 EDATA JMP XEC4A,I
 1618 23573 066574 EILET JMP LTEX
 1619 23574 167604 WREX JMP RITE,I
 1620*
 1621 23575 167605 PREX JMP PRINT,I
 1622*
 1623 23576 167606 DSPEX JMP DISP,I
 1624*
 1625*
 1626*
 1627*
 1628*

LINKS THAT MOVE WITH MONIT

1630*
 1631 23577 010202 EGOS1 OCT 10202
 1632 23600 012233 VAR1 OCT 12233
 1633 23601 004171 STMTA OCT 4171
 1634 23602 016524 SALT OCT 16524
 1635 23603 023773 TABA OCT 23773
 1636 23604 007072 RITE OCT 7072
 1637 23605 006745 PRINT OCT 6745
 1638 23606 006746 DISP OCT 6746
 1639 23607 012436 STRID OCT 12436
 1640 23610 012455 STRI2 OCT 12455
 1641 23611 010437 SWITH OCT 10437
 1642 23612 004117 REEDX OCT 4117
 1643 23613 004120 REEDY OCT 4120
 1644*
 1645*
 1646*

PAGE 351

STRING OPTION BLOCK

| | | | | | | |
|------|-------|--------|-------|-----|--------------------------|--------------|
| 1647 | 23614 | 176660 | | ABS | EVAL-* | |
| 1648 | 23615 | 176566 | | ABS | EINDX-* | |
| 1649 | 23616 | 176545 | | ABS | ELEN-* | |
| 1650 | 23617 | 046511 | ERIL | ASC | 11,MISSING COMMA, RETYPE | |
| | 23620 | 051523 | | | | |
| | 23621 | 044516 | | | | |
| | 23622 | 043440 | | | | |
| | 23623 | 041517 | | | | |
| | 23624 | 046515 | | | | |
| | 23625 | 040454 | | | | |
| | 23626 | 020122 | | | | |
| | 23627 | 042524 | | | | |
| | 23630 | 054520 | | | | |
| | 23631 | 042440 | | | | |
| 1651 | 23632 | 041101 | ER2L | ASC | 8,BAD | STRING INPUT |
| | 23633 | 042040 | | | | |
| | 23634 | 051524 | | | | |
| | 23635 | 051111 | | | | |
| | 23636 | 047107 | | | | |
| | 23637 | 020111 | | | | |
| | 23640 | 047120 | | | | |
| | 23641 | 052524 | | | | |
| 1652 | 23642 | 000054 | B54 | OCT | 54 | |
| 1653 | 23643 | 000021 | .17 | DEC | 17 | |
| 1654 | 23644 | 000377 | NTAB | OCT | 377 | |
| 1655 | 23645 | 001645 | PTR1A | DEF | PTR1 | |
| 1656 | 23646 | 046105 | FTAB | ASC | 1,LE | |
| 1657 | 23647 | 047230 | | OCT | 47230 | N, OPCODE 30 |
| 1658 | 23650 | 050117 | | ASC | 1,PO | |
| 1659 | 23651 | 051631 | | OCT | 51631 | S, OPCODE 31 |
| 1660 | 23652 | 053101 | | ASC | 1,VA | |
| 1661 | 23653 | 046332 | | OCT | 46332 | L, OPCODE 32 |

SYNTAX JUMP TABLE

| | | | | | | |
|-------|-------|--------|--|-----|---------|-------|
| 1662* | | | | | | |
| 1663* | | | | | | |
| 1664* | | | | | | |
| 1665* | | | | | | |
| 1666* | | | | | | |
| 1667* | | | | | | |
| 1668* | | | | | | |
| 1669 | 23654 | 176141 | | ABS | LETS-* | |
| 1670 | 23655 | 176425 | | ABS | READS-* | |
| 1671 | 23656 | 176424 | | ABS | READS-* | INPUT |
| 1672 | 23657 | 176442 | | ABS | DATAS-* | |
| 1673 | 23660 | 176330 | | ABS | PRINS-* | DISP |
| 1674 | 23661 | 176233 | | ABS | IFSYN-* | |
| 1675 | 23662 | 176322 | | ABS | WRTS- | |
| 1676 | 23663 | 176325 | | ABS | PRINS-* | |
| 1677 | 23664 | 176263 | | ABS | IMPLT-* | |

SYNTAX TABLE

| | | | | | | |
|-------|-------|--------|-----|-----|--------|------------|
| 1678* | | | | | | |
| 1679* | | | | | | |
| 1680* | | | | | | |
| 1681 | 23665 | 051105 | SYN | ASC | 2,READ | |
| | 23666 | 040504 | | | | |
| 1682 | 23667 | 104111 | | OCT | 104111 | OPCODE 8,J |
| 1683 | 23670 | 047120 | | ASC | 2,NPUT | |
| | 23671 | 052524 | | | | |

468

PAGE 352

STRING OPTION BLOCK

| | | | | | |
|-------|-------|--------|-------|--------|----------------------|
| 1684 | 23672 | 103504 | OCT | 103504 | OPCODE 7,D |
| 1685 | 23673 | 040524 | ASC | 1,AT | |
| 1686 | 23674 | 040606 | OCT | 40606 | A, OPCODE 6 |
| 1687 | 23675 | 000241 | OCT | 241 | IMPLIED LET |
| 1688 | 23676 | 042111 | ASC | 2,DISP | |
| | 23677 | 051520 | | | |
| 1689 | 23700 | 122511 | OCT | 122511 | OPCODE 5,I |
| 1690 | 23701 | 043204 | OCT | 43204 | F, OPCODE 4 |
| 1691 | 23702 | 053522 | ASC | 2,WRIT | |
| | 23703 | 044524 | | | |
| 1692 | 23704 | 042643 | OCT | 42643 | E, OPCODE 3 |
| 1693 | 23705 | 050122 | ASC | 2,PRIN | |
| | 23706 | 044516 | | | |
| 1694 | 23707 | 052242 | OCT | 52242 | PRINT, OPCODE 2 |
| 1695 | 23710 | 046105 | ASC | 1,LE | LE |
| 1696 | 23711 | 052351 | OCT | 52351 | T, OPCODE 11 |
| 1697* | | | | | |
| 1698* | | | | | |
| 1699* | | | | | |
| 1700 | 23712 | 003626 | ER3A | ABS | ***+2-ID1-ID1 |
| 1701 | 23713 | 041101 | | ASC | 9,BAD NUMERIC INPUT, |
| | 23714 | 042040 | | | |
| | 23715 | 047125 | | | |
| | 23716 | 046505 | | | |
| | 23717 | 051111 | | | |
| | 23720 | 041440 | | | |
| | 23721 | 044516 | | | |
| | 23722 | 050125 | | | |
| | 23723 | 052054 | | | |
| 1702 | 23724 | 003453 | ER4A | ABS | ER1L+ER1L+13-ID1-ID1 |
| 1703 | 23725 | 003436 | ER1A | ABS | ER1L+ER1L-ID1-ID1 |
| 1704 | 23726 | 003464 | ER2A | ABS | ER2L+ER2L-ID1-ID1 |
| 1705 | 01640 | | CP0 | EQU | ST |
| 1706 | 01641 | | CP1 | EQU | ST+1 |
| 1707 | 00160 | | B174K | EQU | M1024 |
| 1708 | 23727 | 000131 | QMRKA | DEF | .63+.63+1 |
| 1709 | 01644 | | INTMP | EQU | ST+4 |
| 1710 | 23730 | 174716 | LENA | ABS | LEN+LEN-ID-ID |
| 1711 | 01645 | | PTR1 | EQU | ST+5 |
| 1712 | 01646 | | PTR2 | EQU | ST+6 |
| 1713 | 01647 | | TMPRE | EQU | ST+7 |
| 1714 | 01742 | | TMP2 | EQU | XC+2 |
| 1715 | 01640 | | TEPPP | EQU | ST |
| 1716 | 01641 | | SPRDD | EQU | ST+1 |
| 1717 | 01643 | | SAVE | EQU | ST+3 |
| 1718 | 01651 | | TEMP6 | EQU | ST+9 |
| 1719 | 01652 | | MPT | EQU | ST+10 |
| 1720 | 01653 | | NQT | EQU | ST+11 |
| 1721 | 01654 | | TNULL | EQU | ST+12 |
| 1722 | 01655 | | PS0 | EQU | ST+13 |
| 1723 | 01656 | | PS1 | EQU | ST+14 |
| 1724 | 23731 | 001742 | TMP2A | DEF | TMP2 |
| 1725 | 23732 | 011761 | MSDAT | OCT | 11761 |
| 1726 | 23733 | 003772 | STDAT | OCT | 3772 |
| 1727 | 01462 | | REEDL | EQU | 1462B |
| 1728* | | | | | |

489

456

1,444,141

456

PAGE 353

STRING OPTION BLOCK

1729*
1730* 23762
1731*
1732*
1733*
1734*
1735 23762 176612
1736 23763 177302
1737 23764 176714
1738 23765 177605
1739 23766 177610
1740 23767 177202
1741 23770 177604
1742 23771 177604
1743 23772 177601
1744*
1745*
1746 23773 176353
1747 23774 177652
1748 23775 177647
1749 23776 177667
1750 23777 004000 ID
1751

* NO ERRORS*

ORG 23762B

STATEMENT EXECUTION TABLE

ABS LTEX-*
ABS EREAD-*
ABS EINPT-*
ABS EDATA-*
ABS DSPEX-*
ABS IFEX-*
ABS WREX-*
ABS PREX-*
ABS EILET-*

ABS LEN-*
ABS FTAB-*
ABS NTAB-*
ABS SYN-*
OCT 4000
END

CROSS-REFERENCE TABLE

[illegible]

PAGE 355

CROSS-REFERENCE TABLE

[illegible]

PAGE 356

CROSS-REFERENCE TABLE

| | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|
| CKMI | 0908 | | | | | | | | |
| CKMOD | 0904 | 0841 | 0928 | | | | | | |
| CLAR2 | 0489 | | | | | | | | |
| CLERA | 0160 | 1013 | 1093 | 1142 | 1153 | 1160 | 1165 | 1248 | 1283 |
| CLINE | 0316 | | | | | | | | |
| CLPIA | 0217 | | | | | | | | |
| COMCE | 0434 | | | | | | | | |
| COMCK | 0433 | 0783 | 0803 | 0918 | | | | | |
| COMP2 | 1299 | 1312 | | | | | | | |
| COMP3 | 1303 | 1300 | | | | | | | |
| COMP4 | 1306 | 1302 | | | | | | | |
| COMP5 | 1309 | 1317 | 1324 | | | | | | |
| COMP6 | 1314 | 1307 | | | | | | | |
| COMP7 | 1318 | 1315 | | | | | | | |
| CONST | 0482 | 0800 | 0955 | 1121 | | | | | |
| CP0 | 1705 | 0867 | 0875 | 0893 | 1295 | 1299 | | | |
| CP1 | 1706 | 0879 | 0882 | 1298 | 1314 | 1319 | | | |
| CPLAC | 0331 | | | | | | | | |
| CRCKA | 0228 | | | | | | | | |
| CRLF | 0495 | | | | | | | | |
| CRLF1 | 0496 | | | | | | | | |
| CRTN | 0037 | 0038 | 0039 | | | | | | |
| CTYPE | 0330 | | | | | | | | |
| CXEC4 | 1165 | 1128 | | | | | | | |
| DATA1 | 0806 | 0801 | | | | | | | |
| DATA2 | 0803 | 0811 | | | | | | | |
| DATAS | 0798 | 0805 | 1672 | | | | | | |
| DCOMA | 0210 | | | | | | | | |
| DELSA | 0175 | | | | | | | | |
| DFLAG | 0388 | 0622 | | | | | | | |
| DIGCK | 0461 | 0610 | | | | | | | |
| DISP | 1638 | 1623 | | | | | | | |
| DPERR | 0043 | | | | | | | | |
| DRQST | 0317 | 1107 | 1120 | 1124 | 1264 | 1271 | 1277 | 1281 | |
| DSPEX | 1623 | 1739 | | | | | | | |
| DSTRT | 0303 | 1364 | 1380 | 1414 | | | | | |
| E | 0080 | | | | | | | | |
| E1 | 0235 | | | | | | | | |
| E10 | 0244 | | | | | | | | |
| E15 | 0249 | 0919 | | | | | | | |
| E20 | 0254 | 0554 | 0653 | 0809 | 0922 | | | | |
| E25 | 0259 | 0792 | | | | | | | |
| E30 | 0264 | 0545 | | | | | | | |
| E35 | 0269 | | | | | | | | |
| E40 | 0274 | 1472 | 1612 | | | | | | |
| E45 | 0279 | | | | | | | | |
| E5 | 0239 | 0692 | 0717 | | | | | | |
| E50 | 0284 | 1404 | | | | | | | |
| ECALA | 0215 | | | | | | | | |
| EDATA | 1617 | 1738 | | | | | | | |
| EGOS1 | 1631 | 1335 | 1345 | | | | | | |
| EILET | 1618 | 1743 | | | | | | | |
| EIN14 | 1203 | 1253 | | | | | | | |
| EIN15 | 1242 | 1236 | | | | | | | |
| EIN16 | 1237 | 1241 | | | | | | | |
| EIN18 | 1224 | 1234 | | | | | | | |

| | | | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|--|--|
| FNDIA | 0224 | | | | | | | | | | | | | |
| FNDAA | 0223 | | | | | | | | | | | | | |
| FNDPS | 0474 | | | | | | | | | | | | | |
| FOPBS | 0206 | | | | | | | | | | | | | |
| FORMA | 0172 | 1115 | 1362 | | | | | | | | | | | |
| FORME | 0336 | | | | | | | | | | | | | |
| FORMS | 0335 | | | | | | | | | | | | | |
| FORMT | 0334 | | | | | | | | | | | | | |
| FPAGE | 0449 | | | | | | | | | | | | | |
| FROMA | 0122 | | | | | | | | | | | | | |
| FSCA | 0173 | 0587 | 0591 | 0758 | 0769 | | | | | | | | | |
| FSCE2 | 0439 | | | | | | | | | | | | | |
| FSCE4 | 0444 | | | | | | | | | | | | | |
| FSCH | 1348 | 1031 | | | | | | | | | | | | |
| FTAB | 1656 | 1747 | | | | | | | | | | | | |
| FTMP1 | 0337 | | | | | | | | | | | | | |
| FTMP2 | 0338 | | | | | | | | | | | | | |
| FWAM | 0157 | | | | | | | | | | | | | |
| FWUP | 0296 | | | | | | | | | | | | | |
| FXFLA | 0194 | 0842 | | | | | | | | | | | | |
| GAI | 1455 | 1450 | 1463 | | | | | | | | | | | |
| GA2 | 1451 | 1446 | | | | | | | | | | | | |
| GADD | 1437 | 0511 | 0527 | 0986 | 0998 | 1181 | 1377 | | | | | | | |
| GETAD | 0459 | | | | | | | | | | | | | |
| GETCH | 0469 | 0878 | 0881 | 0944 | 1061 | 1073 | 1304 | 1322 | 1355 | | | | | |
| GETSK | 0473 | 0734 | 1131 | 1237 | 1242 | | | | | | | | | |
| GFLAG | 0402 | | | | | | | | | | | | | |
| GFRST | 0472 | 0604 | 1204 | 1269 | | | | | | | | | | |
| GNEXT | 0471 | 0605 | 0668 | 0970 | 1201 | 1226 | | | | | | | | |
| GTOOE | 1335 | 1331 | | | | | | | | | | | | |
| GPI | 1004 | 0912 | 0993 | | | | | | | | | | | |
| GP2 | 0997 | 0913 | | | | | | | | | | | | |
| GPARM | 0422 | | | | | | | | | | | | | |
| GPTRS | 0986 | 0866 | 1007 | 1293 | | | | | | | | | | |
| GQUOT | 1592 | 1004 | 1389 | | | | | | | | | | | |
| GSIGN | 0484 | | | | | | | | | | | | | |
| GSYMB | 1611 | 1438 | | | | | | | | | | | | |
| GTCON | 0498 | | | | | | | | | | | | | |
| GTE | 0827 | 0659 | 0659 | | | | | | | | | | | |
| GTEA | 0659 | 0636 | | | | | | | | | | | | |
| GTMP | 0470 | | | | | | | | | | | | | |
| GTOPP | 0464 | 1437 | | | | | | | | | | | | |
| GTSTA | 0213 | 0557 | 0648 | 0740 | 0810 | 0859 | | | | | | | | |
| GVAR | 0966 | 0849 | 0915 | | | | | | | | | | | |
| HFLG1 | 0321 | | | | | | | | | | | | | |
| HSTPT | 0302 | 0940 | 1149 | 1150 | | | | | | | | | | |
| I | 0067 | 0951 | | | | | | | | | | | | |
| ICNT | 0420 | | | | | | | | | | | | | |
| ID | 1750 | 0658 | 0658 | 0659 | 0659 | 1710 | 1710 | | | | | | | |
| ID1 | 0509 | 1162 | 1163 | 1700 | 1700 | 1702 | 1702 | 1703 | 1703 | 1704 | 1704 | | | |
| IF1 | 0644 | | | | | | | | | | | | | |
| IF2 | 0649 | 0647 | | | | | | | | | | | | |
| IF3 | 0648 | 0645 | | | | | | | | | | | | |
| IFERR | 0656 | 0640 | 0646 | | | | | | | | | | | |
| IFEX | 1293 | 1740 | | | | | | | | | | | | |
| IFSUN | 0633 | 1674 | | | | | | | | | | | | |

475

462

1,444,141

462

PAGE 359

CROSS-REFERENCE TABLE

| | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|--|
| ILET1 | 0680 | 0671 | 0674 | 0675 | 0678 | | | | | | |
| ILET2 | 0690 | 0688 | | | | | | | | | |
| IMPLT | 0665 | 1677 | | | | | | | | | |
| IMPY | 0485 | | | | | | | | | | |
| INC1 | 1281 | 1268 | 1270 | | | | | | | | |
| INC2 | 1285 | 1282 | | | | | | | | | |
| INCAL | 1263 | 1180 | 1203 | 1284 | | | | | | | |
| INCH1 | 1160 | 1132 | | | | | | | | | |
| INCH2 | 1153 | 1135 | | | | | | | | | |
| INCH5 | 1142 | 1122 | | | | | | | | | |
| INCHT | 1147 | 1162 | | | | | | | | | |
| INDS2 | 0918 | 0925 | | | | | | | | | |
| INDXS | 0915 | 0865 | | | | | | | | | |
| INF | 0077 | | | | | | | | | | |
| INITS | 0454 | | | | | | | | | | |
| INLN1 | 1162 | 1118 | | | | | | | | | |
| INLNK | 1163 | 1105 | | | | | | | | | |
| INREA | 0226 | | | | | | | | | | |
| INRET | 1108 | 1163 | | | | | | | | | |
| INSTR | 1176 | 1104 | | | | | | | | | |
| INTCK | 0457 | | | | | | | | | | |
| INTEA | 0178 | | | | | | | | | | |
| INTFL | 0075 | | | | | | | | | | |
| INTGR | 0400 | | | | | | | | | | |
| INTMP | 1709 | 0874 | 0892 | 0989 | 1200 | 1217 | 1246 | 1327 | 1400 | 1415 | |
| IOBUF | 0357 | | | | | | | | | | |
| IOHEA | 0229 | 0703 | | | | | | | | | |
| IOSTA | 0225 | | | | | | | | | | |
| KEYFG | 0324 | | | | | | | | | | |
| L2T1 | 0345 | | | | | | | | | | |
| L2T2 | 0346 | 0530 | | | | | | | | | |
| LASTN | 0396 | | | | | | | | | | |
| LBOP | 0119 | 0583 | 1449 | | | | | | | | |
| LDEF | 0487 | | | | | | | | | | |
| LEN | 0824 | 1710 | 1710 | 1746 | | | | | | | |
| LENA | 1710 | | | | | | | | | | |
| LENER | 0900 | 0850 | 0851 | 0916 | 0917 | | | | | | |
| LENS | 0849 | 0907 | | | | | | | | | |
| LENS1 | 0852 | 0860 | 0923 | | | | | | | | |
| LENS2 | 0858 | 0921 | | | | | | | | | |
| LETCK | 0462 | 0670 | | | | | | | | | |
| LETS | 0539 | 1669 | | | | | | | | | |
| LIMIT | 0463 | | | | | | | | | | |
| LISTA | 0167 | | | | | | | | | | |
| LNFD | 0072 | | | | | | | | | | |
| LNINC | 0315 | | | | | | | | | | |
| LNUM | 0455 | 0654 | | | | | | | | | |
| LOCLI | 0343 | 1407 | | | | | | | | | |
| LOCL2 | 0344 | | | | | | | | | | |
| LPCK | 0435 | 0581 | | | | | | | | | |
| LPCKE | 0436 | 0973 | | | | | | | | | |
| LPOP | 0120 | | | | | | | | | | |
| LROMA | 0123 | | | | | | | | | | |
| LSTAK | 0299 | | | | | | | | | | |
| LSTPT | 0301 | 0935 | 1182 | 1503 | 1505 | | | | | | |
| LT2 | 1017 | 1010 | 1012 | | | | | | | | |

CROSS-REFERENCE TABLE

[illegible]

CROSS-REFERENCE TABLE

[illegible]

PAGE 362

CROSS-REFERENCE TABLE

| | | | | | | | | | | | | |
|--------|------|------|------|------|------|------|------|------|------|------|------|--|
| PSTR1 | 1526 | 1522 | | | | | | | | | | |
| PSTR2 | 1530 | 1520 | 1525 | | | | | | | | | |
| PSTR3 | 1562 | 1545 | | | | | | | | | | |
| PSTR4 | 1567 | 1558 | 1561 | 1563 | | | | | | | | |
| PSTR5 | 1570 | 1555 | | | | | | | | | | |
| PSTR6 | 1577 | 1588 | | | | | | | | | | |
| PSTR7 | 1584 | 1574 | | | | | | | | | | |
| PSTR8 | 1537 | 1534 | | | | | | | | | | |
| PTR1 | 1711 | 0885 | 0931 | 1002 | 1015 | 1024 | 1039 | 1040 | 1062 | 1067 | 1198 | |
| | 1206 | 1391 | 1655 | | | | | | | | | |
| PTR1A | 1655 | 0877 | 0943 | 1060 | 1303 | 1354 | | | | | | |
| PTR2 | 1712 | 0852 | 0868 | 0884 | 0939 | 0953 | 0967 | | | | | |
| QMRKA | 1708 | 1265 | | | | | | | | | | |
| QTCHA | 0230 | | | | | | | | | | | |
| QTOP | 0109 | 0161 | | | | | | | | | | |
| QUOCK | 0556 | 0553 | | | | | | | | | | |
| RBOP | 0117 | 0593 | 0756 | | | | | | | | | |
| RBUFF | 0358 | | | | | | | | | | | |
| RCOUT | 0333 | | | | | | | | | | | |
| RDSTR | 1377 | 1361 | | | | | | | | | | |
| RDYDA | 0166 | | | | | | | | | | | |
| READ1 | 0786 | 0782 | | | | | | | | | | |
| READ2 | 0783 | 0788 | 0789 | | | | | | | | | |
| READ3 | 0790 | 0781 | | | | | | | | | | |
| READS | 0779 | 0785 | 1670 | 1671 | | | | | | | | |
| REED | 0161 | | | | | | | | | | | |
| REEDL | 1727 | 1267 | | | | | | | | | | |
| REEDX | 1642 | 1285 | | | | | | | | | | |
| REEDY | 1643 | 1275 | | | | | | | | | | |
| RES | 0415 | | | | | | | | | | | |
| RESB1 | 0451 | | | | | | | | | | | |
| RESBP | 0452 | 0680 | | | | | | | | | | |
| RET2 | 0419 | | | | | | | | | | | |
| RETN2 | 0418 | 0533 | 0578 | 0742 | 0771 | 0774 | | | | | | |
| RETN3 | 0417 | 0582 | 0597 | 0857 | | | | | | | | |
| RITE | 1636 | 1619 | | | | | | | | | | |
| RPCK | 0437 | | | | | | | | | | | |
| RPCKE | 0438 | 0592 | 0704 | 0854 | | | | | | | | |
| RPOP | 0115 | | | | | | | | | | | |
| RSAVE | 0292 | | | | | | | | | | | |
| RSLTE | 0371 | | | | | | | | | | | |
| RSPTR | 0416 | 0906 | 1090 | | | | | | | | | |
| RSTAK | 0414 | | | | | | | | | | | |
| RSTKA | 0158 | 1089 | | | | | | | | | | |
| RESULT | 0424 | 1125 | 1372 | | | | | | | | | |
| RUNA | 0163 | | | | | | | | | | | |
| S | 0082 | | | | | | | | | | | |
| SALT | 1634 | 0625 | | | | | | | | | | |
| SALTA | 0207 | | | | | | | | | | | |
| SAVBP | 0450 | 0672 | | | | | | | | | | |
| SAVE | 1717 | 1106 | 1119 | 1172 | 1259 | 1260 | | | | | | |
| SAVEA | 0325 | | | | | | | | | | | |
| SAVEB | 0326 | | | | | | | | | | | |
| SAVEP | 0815 | 0539 | 0573 | 0665 | 0709 | 0779 | 0798 | | | | | |
| SBADR | 0392 | | | | | | | | | | | |
| SBPTR | 0385 | 0556 | 0584 | 0594 | 0595 | 0596 | 0615 | 0641 | 0643 | 0682 | 0683 | |

CROSS-REFERENCE TABLE

| | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|
| | 0684 | 0706 | 0739 | 0744 | 0757 | 0773 | 0799 | 0858 | 0972 | 1117 | |
| SBPUD | 0429 | 0755 | | | | | | | | | |
| SBUFA | 0199 | 0608 | 0613 | | | | | | | | |
| SEC | 0320 | | | | | | | | | | |
| SFLAG | 0391 | 0621 | | | | | | | | | |
| SGNS | 0491 | | | | | | | | | | |
| SIGN | 0397 | 0398 | | | | | | | | | |
| SLCOD | 0328 | | | | | | | | | | |
| SLEND | 0202 | | | | | | | | | | |
| SLWST | 0445 | 0930 | 1442 | 1452 | 1454 | 1476 | | | | | |
| SNFLA | 0197 | | | | | | | | | | |
| SPLFA | 0196 | | | | | | | | | | |
| SPRDD | 1716 | 1096 | 1097 | 1110 | 1227 | | | | | | |
| SSBPT | 0393 | | | | | | | | | | |
| SSYMA | 0170 | 1611 | | | | | | | | | |
| ST | 0375 | 0579 | 0618 | 0623 | 0637 | 0638 | 0650 | 0651 | 0666 | 0690 | 0712 |
| | 0715 | 0762 | 0767 | 0790 | 0806 | 0816 | 0817 | 0819 | 0856 | 0968 | 1320 |
| | 1333 | 1705 | 1706 | 1709 | 1711 | 1712 | 1713 | 1715 | 1716 | 1717 | 1718 |
| | 1719 | 1720 | 1721 | 1722 | 1723 | | | | | | |
| START | 0162 | 0007 | | | | | | | | | |
| STDAT | 1726 | 1410 | | | | | | | | | |
| STEXP | 0441 | 0639 | 0652 | | | | | | | | |
| STMTA | 1633 | 0627 | | | | | | | | | |
| STORI | 0426 | | | | | | | | | | |
| STORE | 0425 | | | | | | | | | | |
| STPFL | 0323 | | | | | | | | | | |
| STRI2 | 1640 | 0580 | | | | | | | | | |
| STRID | 1639 | | | | | | | | | | |
| STROP | 0428 | | | | | | | | | | |
| STSRH | 0440 | | | | | | | | | | |
| SUBEV | 1470 | 1455 | 1461 | | | | | | | | |
| SUMPA | 0227 | | | | | | | | | | |
| SVARO | 0574 | 0510 | 0540 | 0552 | 0633 | 0644 | 0673 | 0763 | 0780 | 0920 | 0975 |
| SWITH | 1641 | 1363 | 1366 | 1378 | 1395 | | | | | | |
| SYE12 | 0483 | | | | | | | | | | |
| SYE25 | 0458 | | | | | | | | | | |
| SYMC1 | 0431 | 0543 | 0588 | 0676 | | | | | | | |
| SYMC2 | 0430 | 0729 | | | | | | | | | |
| SYMCK | 0432 | | | | | | | | | | |
| SYMTP | 0298 | | | | | | | | | | |
| SYN | 1681 | 1749 | | | | | | | | | |
| SYTSA | 0209 | | | | | | | | | | |
| TABA | 1635 | 0750 | | | | | | | | | |
| TEMP | 0348 | 0349 | 0617 | | | | | | | | |
| TEMP1 | 0350 | | | | | | | | | | |
| TEMP2 | 0351 | | | | | | | | | | |
| TEMP3 | 0352 | | | | | | | | | | |
| TEMP4 | 0353 | | | | | | | | | | |
| TEMP6 | 1718 | 0855 | 0969 | 1231 | 1233 | 1512 | 1523 | 1538 | 1546 | 1577 | 1580 |
| | 1584 | 1597 | 1601 | 1603 | | | | | | | |
| TEMPS | 0347 | 0843 | 0853 | 0908 | 0909 | 0966 | 0987 | 0988 | 0990 | 0996 | 0997 |
| | 1094 | 1099 | 1101 | 1113 | 1126 | 1136 | 1148 | 1358 | 1374 | 1379 | 1382 |
| | 1383 | 1405 | 1443 | 1444 | 1447 | 1456 | 1457 | 1462 | 1475 | 1592 | 1595 |
| | 1602 | | | | | | | | | | |
| TEPPP | 1715 | 1100 | 1114 | 1129 | 1147 | | | | | | |
| TERM | 0362 | 1091 | | | | | | | | | |

PAGE 364

CROSS-REFERENCE TABLE

[illegible]

PAGE 365

9830—BASE PAGE READ-WRITE MEMORY

```

0507
0508*
0509*   R/W TEMPORARIES
0510*
0511   01640   ORG 1640B
0512*
0513   01640   000000   YH   BSS 1
0514   01641   000000   XH   BSS 1
0515   01642   000000   XW   BSS 1
0516   01643   000000   YW   BSS 1
0517   01644   000000   CP3  BSS 1
0518   01645   000000   CP4  BSS 1
0519   01646   000000   T3   BSS 1
0520   01647   000000   T1   BSS 1
0521*
0522   01650   000000   XN   BSS 4
0523   01654   000000   XX   BSS 4
0524   01660   000000   YN   BSS 4
0525   01664   000000   YX   BSS 4
0526*
0527   01670   000000   Z1   BSS 4
0528   01674   000000   Z2   BSS 4
0529*
0530*   THESE FOUR WORDS ARE NOT SAFE DURING FUNCTIONS
0531*
0532   01734   ORG 1734B
0533   01734   000000   DX   BSS 1
0534   01735   000000   DY   BSS 1
0535   01736   000000   CPX  BSS 1
0536   01737   000000   CPY  BSS 1
0537*
0538   01473   CCNT   EQU TEMPS+1
0539   01477   TEMPS  EQU TEMPS+5
0540   01734   CP2    EQU DX
0541   01735   CPT    EQU DY
0542   01735   T4     EQU CPX
0543   01737   T2     EQU CPY

```

9830 PLOTTER BLOCK

```

0545      22000      ORG 22000B
0546*
0547*
0548*      THE PLOT BLOCK HAS ABSOLUTE ADDRESSES TO THE
0549*      FOLLOWING BASIC SYSTEM SUBROUTINES
0550*
0551*      COS
0552*      SIN
0553*      RITE  WRITE STATEMENT SYNTAX
0554*      PRINN  PRINT STATEMENT EXECUTION
0555*      SCERI  FORMAT EXECUTION
0556*      MOVCK  MOVE THE SYMBOL TO LOWER MEMORY
0557*      TYPEC
0558*      QTCHK
0559*      FSPR
0560*      CKEOL
0561*      PFORM
0562*      KEYIN
0563*      VALUP
0564*      TBSRI
0565*
0566*      SPECIAL SYMBOL CODE'S USED IN THIS BLOCK ARE AS FOLLOWS:
0567*
0568*      B1740 SCALING INFORMATION
0569*      B1760 CHARACTER DEFINITION
0570*      1660 CARRIAGE RETURN INFORMATION
0571*
0572*      CODE STARTED 10—26—71
0573*
0574*      RELATIVE ADDRESSES TO SCALING INFORMATION IS AS FOLLOWS
0575*
0576*      0 = X SCALE FACTOR
0577*      4 = Y SCALE FACTOR
0578*      8 = X ADJUST
0579*      12 = Y ADJUST
0580*      16 = X PRESENT POSITION
0581*      20 = Y PRESENT POSITION
0582*      24 = X OFFSET
0583*      28 = Y OFFSET
0584*
0585*      CONSTANTS AND EQUATES
0586*
0587*
0588      22000  001400  N9/6      OCT  1400      1666.66  ** DON'T MOVE — IT'S RELATIVE
                                                ADDRESS = 0
0589      22001  013146              OCT  13146
0590      22002  063146              OCT  63146
0591      22003  063146              OCT  63146
0592*
0593      22004  001400              OCT  1400      9999  ** DON'T MOVE, IT'S RELATIVE
                                                ADDRESS = 4
0594      22005  114631              OCT  114631
0595      22006  000000              OCT  0
0596      22007  000000              OCT  0
0597*

```

PAGE 367

9830 PLOTTER BLOCK

```

0599*
0600*      GET THE PEN CONTROL PARAMETER
0601*
0602      22010  025654  PNPARG  LDB  XX
0603      22011  074113          SBP  *+2
0604      22012  170402          RET
0605      22013  075750  .P1     SZB  *-1
0606      22014  074251          SLB  PENDN
0607*
0608      22015  062032  PENUP   JSM  STATU
0609      22016  075613          SBP  .P1-1
0610      22017  020017          LDA  .2
0611      22020  066024          JMP  PENDN+3
0612      22021  062032  PENDN   JSM  STATU
0613      22022  075412          SBM  .P1-1
0614      22023  020020          LDA  .3
0615      22024  070246          RAR  6
0616*
0617      22025  062032  BUSY    JSM  STATU
0618      22026  074644          SBL  3
0619      22027  075713          SBP  BUSY
0620      22030  043737          IOR  SC16
0621      22031  065356          JMP  TTY+3
0622      22032  027737  STATU   LDB  SC16
0623      22033  172741          STF  1
0624      22034  176141          OTB  1
0625      22035  172741          STF  1
0626      22036  177241          LIB  1,C
0627      22037  074546          RBR  12
0628      22040  170402          RET
0629*
0630      22041  070742  AMOVE    SAR  16
0631      22042  031477          STA  TEMPS
                                ALTERNATE ENTRY FOR
                                ABINCRMENTAL MOVE
0632*
0633      22043  023500  MOVE     LDA  XMIN
0634      22044  062070          JSM  .M
0635      22045  040107          IOR  B1000
0636      22046  062025          JSM  BUSY
0637      22047  021754          LDA  AR2
0638      22050  050105          AND  B377
0639      22051  040013          IOR  B400
0640      22052  062025          JSM  BUSY
0641      22053  023502          LDA  YMIN
0642      22054  062070          JSM  .M
0643      22055  040013          IOR  B400
0644      22056  062025          JSM  BUSY
0645      22057  021754          LDA  AR2
0646      22060  050105          AND  B377
0647      22061  040013          IOR  B400
0648      22062  041477          IOR  TEMP5
0649      22063  062025          JSM  BUSY
0650*
0651      22064  025442          LDB  STPFL
0652      22065  014040          CPB  31
0653      22066  066554          JMP  EPEN
0654      22067  170402          RET
                                STOP KEY HIT
                                YES
                                NO

```

PAGE 368

9830 PLOTTER BLOCK

0655*
0656 22070 060445 .M JSM XFAR2+1
0657 22071 160225 JSM FLTRA,I
0658 22072 001654 XMAX DEF XX
0659 22073 074117 LDA B
0660 22074 074212 SBM *+4
0661 22075 004161 ADB MAXSN
0662 22076 074112 SBM *+2
0663 22077 023514 LDA 9999
0664 22100 031754 STOR2 STA AR2
0665 22101 070342 SAR 8
0666 22102 170402 RET
0667*
0668* FIND THE ADDRESS OF THE R/W WORDS.
0669* ON RETURN THE ADDRESS OF THE 1ST WORD IS IN B AND T1.
0670*
0671 22103 062015 JSM PENUP
0672 22104 020111 GETRW LDA STEAL LOAD THE CODE WORD
0673 22105 160177 JSM SSYMA,I LOOK FOR IT IN SYMBOL TABLE
0674 22106 066113 JMP *+5 NOT FOUND
0675 22107 004132 GEI ADB M1
0676 22110 074537 LDB B,I
0677 22111 035647 STB T1
0678 22112 170402 RET
0679*
0680 22113 160205 JSM AERRA,I
0681 22114 177660 DEC -80 NO SCALE COMMAND EXECUTED
BEFORE PLOT, ECT
0682*
0683*
0684* TRANSFER ROUTINES
0685 22115 021646 WHERE LDA T3
0686 22116 070352 SAM TXM
0687*
0688 22117 027502 TYM LDB YMIN
0689 22120 064406 JMP XFAR1+1
0690*
0691 22121 027504 TYX LDB YMAX
0692 22122 064406 JMP XFAR1+1
0693*
0694 22123 021646 OWHER LDA T3
0695 22124 071552 SAM TYM SKIP TO TYM IF IN XAXIS MODE
0696 22125 027500 TXM LDB XMIN
0697 22126 064406 JMP XFAR1+1
0698*
0699 22127 026072 TXX LDB XMAX
0700 22130 064406 JMP XFAR1+1
0701*
0702 22131 027772 TEMXX LDB TEMX
0703 22132 064406 JMP XFAR1+1
0704*
0705 22133 021737 LDA T2
0706 22134 160216 JSM FADA,I
0707 22135 025737 TRANS LDB T2
0708 22136 064406 JMP XFAR1+1
0709*

PAGE 369

9830 PLOTTER BLOCK

| | | | | | |
|-------|-------|--------|-----|----------|-----------------------|
| 0711* | | | | | |
| 0712 | 22137 | 000051 | ABS | SPEN-* | LETTER EYNTAX |
| 0713 | 22140 | 000105 | ABS | OFFS-* | CPLOT SYNTAX |
| 0714 | 22141 | 000102 | ABS | SCALE-* | SCALE SYNTAX |
| 0715 | 22142 | 000047 | ABS | SAXIS-* | |
| 0716 | 22143 | 000046 | ABS | SAXIS-* | XAXIS SYNTAX |
| 0717 | 22144 | 000103 | ABS | PLOT-* | PLOT SYNTAX |
| 0718 | 22145 | 000100 | ABS | OFFS-* | |
| 0719 | 22146 | 000042 | ABS | SPEN-* | |
| 0720 | 22147 | 000051 | ABS | LABEL-* | |
| 0721 | 22150 | 000077 | ABS | PLOT-* | |
| 0722 | 22151 | 000000 | BSS | 1 | *** CHECKSUM WORD *** |
| 0723* | | | | | |
| 0724 | 22152 | 044520 | ASC | 1,IP | IPLOT |
| 0725 | 22153 | 046117 | ASC | 1,LO | |
| 0726 | 22154 | 052242 | OCT | 52242 | T |
| 0727 | 22155 | 046101 | ASC | 1,LA | LABEL |
| 0728 | 22156 | 041105 | ASC | 1,BE | |
| 0729 | 22157 | 046243 | OCT | 46243 | L |
| 0730 | 22160 | 050105 | ASC | 1,PE | PEN |
| 0731 | 22161 | 047244 | OCT | 47244 | N |
| 0732 | 22162 | 047506 | ASC | 3,OFFSET | |
| | 22163 | 043123 | | | |
| | 22164 | 042524 | | | |
| 0733 | 22165 | 122520 | OCT | 122520 | P |
| 0734 | 22166 | 046117 | ASC | 1,LO | PLOT |
| 0735 | 22167 | 052246 | OCT | 52246 | T; OP CODE 6 |
| 0736 | 22170 | 054101 | ASC | 1,XA | XAXIS |
| 0737 | 22171 | 054111 | ASC | 1,XI | |
| 0738 | 22172 | 051647 | OCT | 51647 | S |
| 0739 | 22173 | 054501 | ASC | 1,YA | YAXIS |
| 0740 | 22174 | 054111 | ASC | 1,XI | |
| 0741 | 22175 | 051650 | OCT | 51650 | S |
| 0742 | 22176 | 051503 | ASC | 2,SCAL | SCALE; OP CODE 9 |
| | 22177 | 045514 | | | |
| 0743 | 22200 | 042651 | OCT | 42651 | E; OP CODE 9 |
| 0744 | 22201 | 041520 | ASC | 2,CPLO | |
| | 22202 | 046117 | | | |
| 0745 | 22203 | 052252 | OCT | 52252 | T; OP CODE 10 |
| 0746 | 22204 | 046105 | ASC | 3,LETTER | FREE MODE |
| | 22205 | 052124 | | | |
| | 22206 | 042522 | | | |
| 0747 | 22207 | 165400 | OCT | 165400 | OP CODE 11 |
| 0748* | | | | | |
| 0749* | | | | | |
| 0750* | | | | | |

END OF SNYNTAX NAME TABLE

PAGE 370

SYNTAX CODE FOR 9830 PLOTTER

0752* PEN STATEMENT SYNTAX

0753*
0754 22210 065147 SPEN JMP ENDS0755*
0756* AXIS SYNTAX0757*
0758 22211 160202 SAXIS JSM FSCA,I DEMAND AN OPPOSITE ORIGIN
0759 22212 060504 JSM COMCK ?
0760 22213 065151 JMP EOST NO DEMAND EOL
0761 22214 160202 JSM FSCA,I YES, DEMAND A TIC PARAMETER
0762 22215 060504 JSM COMCK
0763 22216 065151 JMP EOST
0764 22217 066245 JMP OFFS0765*
0766* LABEL STATEMENT SYNTAX0767*
0768 22220 060456 LABEL JSM SBPUD UPDATE POINTER
0769 22221 060520 JSM LPCKE-1 DEMAND LEFT PAREN.
0770 22222 061055 JSM GNEXT
0771 22223 060456 JSM SBPUD
0772 22224 060463 JSM SYMC1 CHECK FOR "***"
0773 22225 012012 DEF 12012B
0774 22226 066240 JMP LAB2 NOT FOUND, MUST BE NUMBER
0775 22227 060456 JSM SBPUD
0776 22230 061055 JSM GNEXT
0777 22231 060504 LAB3 JSM COMCK FOUND, NOW CHECK FOR OPTIONAL
PARAMETERS
0778 22232 066237 JMP LAB1 P+1, NO OPTIONS, GO DEMAND A
RIGHT PAREN.
0779 22233 062254 JSM FORM3 P+2, OPTIONS ARE THERE
0780 22234 060504 JSM COMCK CHECK IF PAPER DIMENSIONS ARE
PRESENT
0781 22235 066237 JMP LAB1 NO
0782 22236 160202 JSM FSCA,I YES, GET A FORMULA
0783 22237 167562 LAB1 JMP RITE,I
0784 22240 055706 LAB2 DSZ SBPTR
0785 22241 060663 JSM LNUM-1
0786 22242 066231 JMP LAB30787*
0788* SCALE STATEMENT SYNTAX0789*
0790 22243 062256 SCALE JSM FORM2
0791 22244 060510 JSM COMCE
0792 22245 062256 OFFS JSM FORM2
0793 22246 065151 JMP EOST0794*
0795* PLOT AND IPLOT STATEMENT SYNTAX0796*
0797 22247 062256 PLOT JSM FORM2 DEMAND X AND Y COOR.
0798 22250 060504 JSM COMCK CHECK FOR COMMA
0799 22251 065151 JMP EOST " " NOT FOUND, DEMAND E.O.S.
0800 22252 160202 JSM FSCA,I PROCESS OPTIONAL PEN COMMAND
0801 22253 065151 JMP EOST DEMAND E.O.S.
0802*
0803 22254 160202 FORM3 JSM FSCA,I
0804 22255 060510 JSM COMCE DEMAND,
0805 22256 160202 FORM2 JSM FSCA,I
0806 22257 060510 JSM COMCE DEMAND,
0807 22260 164202 JMP FSCA,I

PAGE 371

EXECUTION PHASE OF 9830 PLOTTER BLOCK

```

0809*
0810 22261 160201 FORM JSM FORMA,I
0811 22262 160201 JSM FORMA,I
0812 22263 160201 JSM FORMA,I
0813 22264 164201 JMP FORMA,I
0814*
0815* SCALE EXECUTION
0816*
0817 00111 STEAL EQU B1740
0818*
0819 22265 062652 ESCAL JSM CURPG GET CURRENT PAGE
0820 22266 062261 JSM FORM STACK 4 FORMULA
0821*
0822 22267 020111 LDA STEAL LOAD SYMBOL FOR R/W WORDS
0823 22270 160177 JSM SSYMA,I CHECH IF PLOTTER WORDS ARE
                                SAVED ALREADY
0824 22271 066273 JMP *+2 NO, ENTER SAME
0825 22272 066306 JMP ESI YES, SKIP ENTRY
0826 22273 025406 LDB VALUE ENTER PLOTTER R/W WORDS INTO
                                STACKS
0827 22274 023520 LDA M34 MOVE DOWN 34 PLACES, 32 FOR
                                VALUES
0828*
0829* MOVE H-STACK DOWN AND CHEKC FOR OVERFLOW
0830*
0831* MOVE DOWN -(A) WORDS
0832* (B)=ADDRESS+1 OF THE LAST WORD TO BE MOVED
0833*
0834 22275 163666 JSM MOVCK,I
0835*
0836 22276 003520 ADA M34
0837 22277 070137 LDB A
0838 22300 000017 ADA .2
0839 22301 074557 STA B,I
                                POINTER TO VALUE TABLE
                                POINTER TO 1ST WORD OF R/W
                                STORE ADDRESS POINTER TO
                                PLOTTER R/W
0840 22302 074070 SIB *+1
0841 22303 020111 LDA STEAL LOAD PLOTTER L/W CODE
0842 22304 074557 STA B,I
0843 22305 035406 STB VALUE
0844*
0845 22306 062107 ESI JSM GEI LOAD THE ADDRESS POINTER TO 1ST
                                WORD OF PLOTTER R/W
0846 22307 035737 STB T2 SAVE B TEMPORARLY FOR ZERING
0847 22310 070742 SAR 16 CLEAR A
0848 22311 027554 LDB M31
0849 22312 131737 STA T2,I ZERO 33 WORDS OF R/W
0850 22313 045737 ISZ T2
0851 22314 077730 RIB *-2 LOOP BACK IF NOT ZERO
0852*
0853*
0854* COMPUTE THE SCALE FACTOR - 9999/(YMAX-YMIN)
0855* AND YADJUST = -(S.F.*YMIN)
0856*
0857 22315 020021 LDA .4 * SET-UP Y SCALE FACTOR
0858 22316 062322 JSM SCAL *
0859 22317 070004 SAL 16 *
0860 22320 062322 JSM SCAL *
0861 22321 164172 JMP XEC4A,I RETURN TO MONITOR
0862*
0863* SCAL ROUTINE IS USED FOR X AND Y SCALE FACTORS
0864* AND ADJUST VALUES

```

PAGE 372

EXECUTION PHASE OF 9830 PLOTTER BLOCK

| | | | | | | |
|-------|-------|--------|------|-----|---------|--------------------------------|
| 0865* | | | | | | |
| 0866 | 22322 | 031735 | SCAL | STA | CPT | |
| 0867 | 22323 | 061220 | | JSM | FETCH+1 | RECALL THE MAX |
| 0868 | 22324 | 062127 | | JSM | TXX | |
| 0869 | 22325 | 061220 | | JSM | FETCH+1 | RECALL THE MIN |
| 0870 | 22326 | 062117 | | JSM | TYM | |
| 0871 | 22327 | 024033 | | LDB | .16 | |
| 0872 | 22330 | 062344 | | JSM | TR/W-1 | |
| 0873 | 22331 | 022072 | | LDA | XMAX | |
| 0874 | 22332 | 160217 | | JSM | .FSBA,I | (MAX-MIN) |
| 0875 | 22333 | 062125 | | JSM | TXM | |
| 0876 | 22334 | 020021 | | LDA | .4 | RELATIVE ADDRESS TO 9999 |
| 0877 | 22335 | 041645 | | IOR | CP4 | INCLUDE CURRENT PAGE |
| 0878 | 22336 | 160221 | | JSM | .FDVA,I | 9999/(MAX-MIN) |
| 0879 | 22337 | 025735 | | LDB | CPT | CPT=POINTER TO FIRST WORD IN |
| | | | | | | R/W FOR X OR Y |
| 0880 | 22340 | 062345 | | JSM | TR/W | PLACE THE S.F. IN THE STACK |
| 0881 | 22341 | 023502 | | LDA | YMIN | |
| 0882 | 22342 | 160220 | | JSM | .FMPI,I | S.F. * MIN |
| 0883 | 22343 | 024025 | | LDB | .8 | SET(B)=ADJUST LOCATION POINTER |
| 0884 | 22344 | 005735 | | ADB | CPT | |
| 0885 | 22345 | 005647 | TR/W | ADB | TI | |
| 0886 | 22346 | 064406 | | JMP | XFAR1+1 | |

PAGE 373

EXECUTION PHASE OF 9830 PLOTTER BLOCK

| OFFSET COMMAND EXECUTION | | | | | | |
|--------------------------|-------|--------|-------|-----|---------|--|
| 0888* | | | | | | |
| 0889* | | | | | | |
| 0890* | | | | | | |
| 0891 | 22347 | 062263 | EOFF | JSM | FORM+2 | |
| 0892 | 22350 | 062104 | | JSM | GETRW | |
| 0893 | 22351 | 023532 | | LDA | B24 | |
| 0894 | 22352 | 062356 | | JSM | EOFF1 | |
| 0895 | 22353 | 020033 | | LDA | .16 | |
| 0896 | 22354 | 062356 | | JSM | EOFF1 | |
| 0897 | 22355 | 164172 | | JMP | XEC4A,I | |
| 0898* | | | | | | |
| 0899 | 22356 | 031735 | EOFF1 | STA | CPT | |
| 0900 | 22357 | 061220 | | JSM | FETCH+1 | |
| 0901 | 22360 | 062343 | | JSM | TR/W-2 | |
| 0902 | 22361 | 074117 | | LDA | B | |
| 0903 | 22362 | 002610 | | ADA | M8 | |
| 0904 | 22363 | 160217 | | JSM | .FSBA,I | |
| 0905 | 22364 | 025735 | | LDB | CPT | |
| 0906 | 22365 | 066345 | | JMP | TR/W | |
| 0907* | | | | | | |

PAGE 374

EXECUTION PHASE OF 9830 PLOTTER BLOCK

```

0909*
0910 22366 021646 PWHER LDA T3 PWHER IS USED BY AXIS EXECUTION
0911 22367 070352 SAM PUNX+1
0912 22370 066372 JMP PUNY+1
0913*
0914* ROUTINE TO GET A PARAMETER AND CONVERT IT TO PLOTTER UNITS
0915*
0916 22371 061220 PUNY JSM FETCH+1
0917 22372 020021 LDA .4
0918 22373 027502 LDB YMIN
0919 22374 066400 JMP PUNIT
0920*
0921* PUNX FETCHES A PARAMETER AND CONVERTS IT TO PLOTTER UNITS
0922*
0923 22375 061220 PUNX JSM FETCH+1
0924 22376 070742 SAR 16 SET A=TO AN OFFSET SO THAT
                                A+T1=1ST WORD
0925 22377 027500 LDB XHIN
0926 22400 001647 PUNIT ADA T1
0927 22401 031734 STA CP2 STORE POINTER TO 1ST X OR Y VALUE
0928 22402 000033 ADA .16
0929 22403 031736 STA T4 SAVE POINTER TO PRESENT POSITION
0930 22404 035737 STB T2 STORE POINTER TO XMIN OR YMIN
0931 22405 060406 JSM XFAR1+1 TRANSFER AR2 TO XMIN OR YMIN
0932 22406 121736 LDA T4,I LOAD THE EXP WORD OF THE
                                PRESENT POSITION
0933 22407 070006 RAR 1
0934 22410 070231 SLA .PUN,C IS BIT 1=1?
0935 22411 070706 RAR 15 YES, CLEAR IT, AND GIVE PENUP
0936 22412 131736 STA T4,I
0937 22413 062015 JSM PENUP
0938 22414 021644 .PUN LDA CP3
0939 22415 070153 SAP PUN1 SKIP IF IN ABS MODE
0940 22416 025736 LDB T4
0941 22417 062133 JSM TRANS-2 X/YMIN-LAST POSITION + DX/DY IF
                                IN INCR.
0942*
0943 22420 025736 PUN1 LDB T4
0944 22421 060406 JSM XFAR1+1 REPLACE PRESENY POSITION WITH
                                NEW VALUE
0945 22422 004025 ADB .8
0946 22423 062133 JSM TRANS-2 NEW POSITION + OFFSET
0947 22424 021734 LDA CP2
0948 22425 160220 JSM .FMPA,I S.F.*NEW POSITION
0949 22426 062135 JSM TRANS
0950 22427 074117 LDA B
0951 22430 025734 LDB CP2
0952 22431 004025 ADB .8
0953 22432 160217 JSM .FSBA,I SUBTRACT THE ADJUST VALVE
0954 22433 062135 JSM TRANS SAVE THIS ANSWER (PLOTTER UNITS)
                                IN XMIN OR YMIN
0955 22434 074742 SBR 16 CLEAR (B) FOR A FLAG IF OUT OF
                                RANGE
0956*
0957 22435 021754 LIMIT LDA AR2
0958 22436 070451 SLA XLT2
0959 22437 021737 LDA T2 WAS THE RESULT NEGATIVE
                                YES — CLEAR X OR Y MIN
0960 22440 170000 CLR
0961 22441 024033 LFLAG LDB .16 FLAG PRESENT PEN POSITION
0962 22442 005734 ADB CP2
0963 22443 074517 LDA B,I
0964 22444 040017 IOR .2

```

PAGE 375

EXECUTION PHASE OF 9830 PLOTTER BLOCK

| | | | | | | |
|-------|-------|--------|------|-----|-------|----------------------------------|
| 0965 | 22445 | 074557 | | STA | B,I | |
| 0966 | 22446 | 170402 | | RET | | |
| 0967* | | | | | | |
| 0968 | 22447 | 003512 | XLT2 | ADA | EXPM4 | THE RESULT WAS NOT NEGATIVE, NOW |
| | | | | | | NOW CHECK IF |
| 0969 | 22450 | 071712 | | SAM | *-2 | <9999? |
| 0970 | 22451 | 020021 | | LDA | .4 | OTHERWISE SET XMIN OR YMIN |
| | | | | | | (RELATIVE ADD. TO 9999) |
| 0971 | 22452 | 041645 | | IOR | CP4 | |
| 0972 | 22453 | 025737 | | LDB | T2 | TO 9999 |
| 0973 | 22454 | 170004 | | XFR | | |
| 0974 | 22455 | 066441 | | JMP | LFLAG | AND FLAG THE LAST POSITION |

PAGE 376

EXECUTION PHASE OF 9830 PLOTTER BLOCK

| | | | | | | |
|-------|-------|--------|-------|-----|---------|---------------------------|
| 0976* | | | | | | |
| 0977 | 22456 | 075410 | XX | SZB | XLT2-1 | |
| 0978 | 22457 | 024133 | | LDB | M2 | |
| 0979 | 22460 | 035654 | | STB | XX | |
| 0980 | 22461 | 066015 | | JMP | PENUP | |
| 0981* | | | | | | |
| 0982* | | | | | | |
| 0983 | 22462 | 076052 | EIPLT | SBM | *+1,S | |
| 0984 | 22463 | 035644 | EPLOT | STB | CP3 | |
| 0985 | 22464 | 074072 | | SBM | *+1,C | |
| 0986 | 22465 | 062652 | | JSM | CURPG | |
| 0987 | 22466 | 062263 | | JSM | FORM+2 | STACK X, AND Y |
| 0988 | 22467 | 021472 | | LDA | TEMPS | |
| 0989 | 22470 | 011421 | | CPA | PRADD | PEN CONTROL? |
| 0990 | 22471 | 066474 | | JMP | *+3 | |
| 0991 | 22472 | 061217 | | JSM | FETCH | |
| 0992 | 22473 | 160225 | | JSM | FLTRA,I | |
| 0993 | 22474 | 024133 | | LDB | M2 | |
| 0994 | 22475 | 035654 | | STB | XX | |
| 0995* | | | | | | |
| 0996 | 22476 | 062104 | | JSM | GETRW | |
| 0997* | | | | | | |
| 0998 | 22477 | 062371 | | JSM | PUNY | FETCH Y |
| 0999 | 22500 | 062456 | | JSM | XX | CHECK IF PEN IS OFF-SCALE |
| 1000 | 22501 | 062375 | | JSM | PUNX | FETCH X |
| 1001 | 22502 | 062456 | | JSM | XX | CHECK IF PEN IS OFF-SCALE |
| 1002 | 22503 | 062010 | | JSM | PNPAR | CHECK FOR ANY PEN CONTROL |
| | | | | | | BEFORE MOVEMENT |
| 1003 | 22504 | 062041 | | JSM | AMOVE | SET ABS FLAG AND MOVE PEN |
| 1004 | 22505 | 025654 | | LDB | XX | LOAD THE PEN CONTROL WORD |
| 1005 | 22506 | 074113 | | SBP | *+2 | |
| 1006 | 22507 | 062013 | | JSM | .P1 | AND CHECK FOR ANY PEN |
| | | | | | | MOVEMENT AFTERWORDS |
| 1007 | 22510 | 164172 | | JMP | XECA,I | |

PAGE 377

EXECUTION PHASE OF 9830 PLOTTER BLOCK

1009*
1010* X AND Y AXIS EXECUTION
1011*
1012 22511 076053 EXAXE SBP *+1,S
1013 22512 035646 EYAXE STB T3
1014 22513 074072 SBM *+1,C
1015 22514 062652 JSM CURPG
1016 22515 035644 STB CP3 SET PUNIT FOR ABS MODE
1017 22516 160201 JSM FORMAI,1 STACK ORIGIN
1018 22517 021472 LDA TEMPS
1019 22520 011421 CPA PRADD ORIGIN ONLY?
1020 22521 066546 JMP AX7
1021 22522 160201 JSM FORMAI,1 STACK TICK
1022 22523 021472 LDA TEMPS
1023 22524 011421 CPA PRADD ORIGIN, TICK ONLY?
1024 22525 066556 JMP AX3 YES
1025 22526 062263 JSM FORM+2 STACK POINT 1 and 2
1026 22527 062103 JSM GETRW-1 DO PENUP AND GET THE R/W ADDRESS
1027 22530 062604 JSM CLIMT-2 RECALL POINT 2
1028 22531 035665 STB YX+1
1029 22532 061220 JSM FETCH+1
1030 22533 062127 JSM TXS
1031 22534 062605 JSM CLIMT-1 SAVE THE STARTING POINT
1032 22535 061220 .EI JSM FETCH+1
1033 22536 062131 JSM TEMXX
1034*
1035* YX = MIN VALUE IN PLOTTER UNITS
1036* YX+1 = MAX VALUE IN PLOTTER UNITS
1037* XMAX HAS PRESENT PEN POSITION
1038* TEMX HAS THE TICK INCREMENT
1039*
1040*
1041 22537 062573 SLOOP JSM AX4
1042 22540 062633 JSM .TICK PUT PEN DOWN AND MAKE A TICK AT THE MIN POINT
1043 22541 022072 AX5 LDA XMAX
1044 22542 027772 LDB TEMX
1045 22543 160216 JSM .FADA,I
1046 22544 062614 JSM TICMK COMPUTE THE NEXT POINT
1047 22545 066541 JMP AX5 GO TO THE TICMK ROUTINE TO CONVERT TO P.U.
1048* MOVE THE PEN AND MAKE A TIC THEN LOOP BACK
1049 22546 062560 AX7 JSM AX2
1050 22547 062573 JSM AX4 NO TICKS OR NIN AND MAX
1051*
1052 22550 025665 AX6 LDB YX+1 TICMK WILL COME HERE IF IT GOES BEYOND MAX VALUE
1053 22551 160227 JSM FXFLAI
1054 22552 062115 JSM WHERE
1055 22553 062602 JSM .AX4
1056 22554 062015 EPEN JSM PENUP
1057 22555 164172 JMP XEC4A,I DO PENUP, PEN EXECUTION ALSO AND RETURN
1058*
1059*
1060* SUBROUTINES FOR AXIS EXECUTION
1061*
1062 22556 062560 AX3 JSM AX2
1063 22557 066535 JMP .EI
1064*

PAGE 378

EXECUTION PHASE OF 9830 PLOTTER BLOCK

| | | | | | | |
|-------|-------|--------|-------|-----|---------|---|
| 1065 | 22560 | 062103 | AX2 | JSM | GETRW-1 | DO PENUP AND GET THE R/W ADDRESS |
| 1066 | 22561 | 023514 | | LDA | .9999 | |
| 1067 | 22562 | 031665 | | STA | YX+1 | (B) = POINTER TO SCALE FACTOR |
| 1068 | 22563 | 021646 | | LDA | T3 | |
| 1069 | 22564 | 070112 | | SAM | *+2 | |
| 1070 | 22565 | 004021 | | ADB | .4 | |
| 1071 | 22566 | 074117 | | LDA | B | |
| 1072 | 22567 | 000025 | | ADA | .8 | |
| 1073 | 22570 | 160221 | | JSM | .FDVA,I | ADJUST / SCALE FACTOR |
| 1074 | 22571 | 062127 | | JSM | TXX | |
| 1075 | 22572 | 066366 | | JMP | PWHER | |
| 1076* | | | | | | |
| 1077 | 22573 | 061220 | AX4 | JSM | FETCH+1 | RECALL THE OPPOSITE ORIGIN |
| 1078 | 22574 | 021646 | | LDA | T3 | |
| 1079 | 22575 | 031664 | | STA | YX | INITIALIZE YX TO A VALUE ABOVE 9999 |
| 1080 | 22576 | 070076 | | TCA | | |
| 1081 | 22577 | 062367 | | JSM | PWHER+1 | |
| 1082 | 22600 | 062606 | | JSM | CLIMT | |
| 1083 | 22601 | 035666 | | STB | YX+2 | |
| 1084 | 22602 | 062041 | AX4 | JSM | AMOVE | |
| 1085 | 22603 | 066021 | | JMP | PENDN | |
| 1086* | | | | | | |
| 1087 | 22604 | 061220 | | JSM | FETCH+1 | |
| 1088 | 22605 | 062366 | | JSM | PWHER | |
| 1089 | 22606 | 076210 | CLIMT | RZB | .CL2 | IF (B) ≠ 0, THEN OUT OF RANGE |
| 1090 | 22607 | 160225 | .CL1 | JSM | FLTRA,I | PLOTTER UNITS DID NOT OVERFLOW, SO |
| 1091 | 22610 | 177770 | M8 | DEC | -8 | |
| 1092 | 22611 | 170402 | | RET | | RETURN AFTER CONVERTING TO INTEGER |
| 1093 | 22612 | 160205 | .CL2 | JSM | AERRA,I | |
| 1094 | 22613 | 177656 | | DEC | -82 | ERROR IF XO, YO, OR THE MIN, MAX VALUES OFF-SCALE |
| 1095* | | | | | | |
| 1096* | | | | | | |
| 1097* | | | | | | |
| 1098* | | | | | | |
| 1099 | 22614 | 062127 | TICK | JSM | TXX | |
| 1100 | 22615 | 062366 | | JSM | PWHER | |
| 1101 | 22616 | 062607 | | JSM | .CL1 | |
| 1102 | 22617 | 015664 | | CPB | YX | DID THE VALUE CHANGE? |
| 1103 | 22620 | 066612 | | JMP | .CL2 | NO, MUST BE A ZERO TICK OR XMAX + TICK = XMAX |
| 1104 | 22621 | 035664 | | STB | YX | STORE A GOOD VALUE FOR THE 1ST INCREMENT |
| 1105 | 22622 | 021670 | | LDA | Z1 | LOAD THE SIGN OF THE TICK |
| 1106 | 22623 | 074076 | | TCB | | |
| 1107 | 22624 | 005665 | | ADB | YX+1 | |
| 1108 | 22625 | 074150 | | SZB | *+3 | DO A TICK IF ZERO |
| 1109 | 22626 | 070111 | | SLA | *+2 | |
| 1110 | 22627 | 076212 | | SBM | .TICK,S | - DIRECTION |
| 1111 | 22630 | 074153 | | SBP | .TICK | + DIRECTION |
| 1112 | 22631 | 055777 | TBL | DSZ | RSPTR | |
| 1113 | 22632 | 066550 | | JMP | AX6 | FINISH THE AXIS AND RETURN |
| 1114 | 22633 | 062041 | TICK | JSM | AMOVE | |
| 1115 | 22634 | 025666 | | LDB | YX+2 | RECALL THE OPPOSITE AXIS |
| 1116 | 22635 | 004052 | | ADB | .48 | |
| 1117 | 22636 | 074117 | | LDA | B | |
| 1118 | 22637 | 000161 | | ADA | MAXSN | (-10000) |
| 1119 | 22640 | 070113 | | SAP | *+2 | |
| 1120 | 22641 | 062647 | | JSM | .TIC | OUTPUT PLUS TICK |

480

480

1,444,141

480

PAGE 379

EXECUTION PHASE OF 9830 PLOTTER BLOCK

| | | | | |
|-------|-------|--------|-----|---------|
| 1121 | 22642 | 024153 | LDB | M48 |
| 1122 | 22643 | 005666 | ADB | YX+2 |
| 1123 | 22644 | 074112 | SBM | *+2 |
| 1124 | 22645 | 062647 | JSM | TIC |
| 1125 | 22646 | 025666 | LDB | YX+2 |
| 1126 | 22647 | 160227 | JSM | FXFLA.I |
| 1127 | 22650 | 062123 | JSM | OWHER |
| 1128 | 22651 | 066041 | JMP | AMOVE |
| 1129* | | | | |

PAGE 380

EXECUTION PHASE OF 9830 PLOTTER BLOCK

```

1131*
1132 22652 074442 CURPG SBR 10
1133 22653 074304 SBL 10
1134 22654 035645 STB CP4
1135 22655 170402 RET
1136*
1137 22656 026072 ROT2 LDB XMAX
1138 22657 160221 JSM .FDVA,I /A.R
1139 22660 062131 ROT4 JSM TEMXX
1140 22661 023504 ROT1 LDA YMAX
1141 22662 160220 JSM .FMFA,I * HEIGHT
1142 22663 160225 JSM FLTRA,I CONVERT TO INTGER
1143 22664 066672 JMP PRERR OVERFLOW RETURN
1144 22665 074117 LDA B
1145 22666 070112 SAM *+2
1146 22667 070076 TCA
1147 22670 000122 ADA B6K
1148 22671 071213 SAP CURPG+3 RETURN
1149 22672 160205 PRERR JSM AERRA,I
1150 22673 177657 DEC -81 CHARACTER SIZE TO BIG, OR BINARY
                                         FORMAT NOT ALLOWED
1151*
1152* SET DEFAULT HEIGHT APPROX = 1.5%, A.R. = 2
1153*
1154 22674 062652 CDEF JSM CURPG
1155 22675 062015 JSM PENUP
1156 22676 020112 LDA B1760
1157 22677 160177 JSM SSYMA,I SERACH THE SYMBOL TABLE FOR CHAR
                                         DEFINITION
1158 22700 066720 JMP .CDEF
1159 22701 062107 JSM GE1 FOUND
1160 22702 035640 STB YH
1161 22703 074070 SIB *+1
1162 22704 035641 STB XH
1163 22705 074070 SIB *+1
1164 22706 035642 STB XW
1165 22707 074070 SIB *+1
1166 22710 035643 STB YW
1167 22711 023502 GETLC LDA PPPOS
1168 22712 160177 JSM SSYMA,I SET UP A LOCATION FOR CRLF'S
1169 22713 066715 JMP *+2
1170 22714 066107 JMP GE1
1171 22715 160252 JSM ENTEA,I
1172 22716 062107 JSM GE1
1173 22717 067376 JMP .PC
1174*
1175 22720 160252 CDEF JSM ENTEA,I
1176 22721 062701 JSM CDEF+5
1177 22722 021640 LDA YH
1178 22723 170000 CLR
1179 22724 020105 LDA B377
1180 22725 131640 STA YH,I
1181 22726 020104 LDA B177
1182 22727 131642 STA XW,I
1183 22730 170402 RET
1184*
1185 22731 020020 ELAB3 LDA ONE
1186 22732 060445 JSM XFAR2+1

```

PAGE 381

EXECUTION PHASE OF 9830 PLOTTER BLOCK

```

1187 22733 066757      JMP .ELAB
1188*
1189*      LETTING SUBROUTINE TO SET-UP CHARACTER DEFINITION
1190*
1191 22734 062652  ELAB  JSM  CURPG  SAVE CURRENT PAGE
1192 22735 045472      ISZ  TEMPS
1193 22736 023516      LDA  FMTOP
1194 22737 031640      STA  1640B  SET OP CODE FOR FORMAT SREACH
1195 22740 074742      SBR  16  SET REPEAT COUNT TO ZERO
1196 22741 163572      JSM  PLOTE,I  GO CHECK FOR FORMAT STATEMENT
1197 22742 074742      SBR  16  P+1, NOT FOUND
1198 22743 035646      STB  T3  P+2, FOUND
1199 22744 062675      JSM  CDEF+1
1200 22745 121472      LDA  TEMPS,I
1201 22746 050065      AND  OPMSK  CHECK IF THERE ARE MORE
                                           PARAMETERS
1202 22747 010004      CPA  RPOP  WAS IT A RIGHT PAREN
1203 22750 067023      JMP  DEFLT
1204*
1205 22751 062262      JSM  FORM+1  STACK 3 FORMULA'S
1206 22752 121472      LDA  TEMPS,I
1207 22753 050065      AND  OPMSK  GET NEXT OPERAND
1208 22754 010004      CPA  RPOP  WAS IT A PAREN
1209 22755 066731      JMP  ELAB3  DEFAULT THE PAPER RATIO
1210 22756 061217      JSM  FETCH  FETCH THE PAPER RATIO
1211 22757 062125  .ELAB JSM  TXM
1212 22760 061220      JSM  FETCH+1  ROTATION
1213 22761 062117      JSM  TYM
1214 22762 061220      JSM  FETCH+1  ASPECT RATIO
1215 22763 062127      JSM  TXX
1216 22764 061220      JSM  FETCH+1  HEIGHT
1217 22765 063352      JSM  .MPY3  REDUCE HEIGHT BY 1/10
1218 22766 062121      JSM  TYX
1219*
1220*      PROCESS CHARACTER DIMENSION
1221*
1222 22767 021645      LDA  CP4
1223 22770 160220      JSM  .FMPI,I
1224 22771 062121      JSM  TYX
1225 22772 023502      LDA  YMIN
1226 22773 163566      JSM  SIN,I
1227 22774 027522      LDB  TEMY
1228 22775 060406      JSM  XFAR1+1
1229 22776 023502      LDA  YMIN
1230 22777 163570      JSM  COS,I
1231 23000 062117      JSM  TYM  YMIN = COS
1232*
1233 23001 062661      JSM  ROT1  YH = 9999/6 * H * COS TH
1234 23002 135640      STB  YH,I
1235*
1236 23003 023522      LDA  TEMY  SIN TH
1237 23004 062656      JSM  ROT2  XH = 9999/6 * H/A.R. * SIN TH
1238 23005 135641      STB  XH,I
1239*
1240 23006 023502      LDA  YMIN  COS TH
1241 23007 027500      LDB  XMIN
1242 23010 160220      JSM  .FMPI,I

```

PAGE 382

EXECUTION PHASE OF 9830 PLOTTER BLOCK

1243 23011 062131 JSM TEMXX
 1244 23012 023772 LDA TEMX
 1245 23013 062656 JSM ROT2 XW = 9999/6 * H/A.R. * COS TH * PH/PW
 1246 23014 135642 STB XW,I
 1247*
 1248 23015 027500 LDB XMIN
 1249 23016 074637 ISZ B,I SET PAPER RATIO NEGATIVE FOR
 -(RESULT)
 1250 23017 023522 LDA TEMY
 1251 23020 160220 JSM FMPI,I YW = 9999/6 * H * SIN TH * PH/PW
 1252 23021 062660 JSM ROT4
 1253 23022 135643 STB YW,I
 1254*
 1255 23023 021646 DEFLT LDA T3
 1256 23024 070110 SZA ELAB6
 1257 23025 067054 JMP WREXC-3
 1258*
 1259* PRINT SYNTAX EXECUTION
 1260*
 1261 23026 025416 ELAB6 LDB MODE
 1262 23027 076150 RZB *+3
 1263 23030 021622 LDA TERM
 1264 23031 072150 RZA *+3
 1265 23032 160170 JSM CLERA,I
 1266 23033 035622 STB TERM
 1267 23034 020146 LDA M15
 1268 23035 031453 STA PRFLG
 1269 23036 163510 JSM CKEOL,I
 1270 23037 067177 JMP ELAB5
 1271 23040 067177 JMP ELAB5
 1272 23041 163574 ELAB1 JSM PRINN,I SET THE BUFFER
 1273 23042 067177 JMP ELAB5 END OF LINE
 1274 23043 021625 ELAB4 LDA FIRST
 1275 23044 070076 TCA
 1276 23045 001400 ADA BADDR
 1277 23046 071550 SZA ELAB1
 1278 23047 025625 LDB FIRST
 1279 23050 063203 JSM OUTCH OUTPUT THE BUFFER
 1280 23051 021400 LDA BADDR
 1281 23052 031625 STA FIRST
 1282 23053 067041 JMP ELAB1
 1283*
 1284* FORMAT TYPE EXECUTION
 1285*
 1286 23054 163510 JSM CKEOL,I
 1287 23055 000153 B153 OCT 153
 1288 23056 035472 STB TEMPS
 1289 23057 163652 WREXC JSM TYPEC,I GET NEXT FORMAT SPEC
 1290 23060 067072 JMP WRITN P+1, END OF FORMAT
 1291 23061 067102 JMP WRITQ P+2 QUOTE
 1292 23062 074742 SBR 16 P+3 FIXED FORMAT
 1293 23063 060563 JSM SLWST P+4 STACK FLAG 0-F; +=E, -=BINARY
 1294 23064 067104 JMP WRITF P+5 (CODE 5 NOT USED)
 1295 23065 066672 JMP PRERR BINARY MODE NOT ALLOWED
 1296 23066 067077 JMP WRITX X;
 1297 23067 163540 JSM FSPR,I
 1298 23070 063370 JSM PCRLF+1; CRLF

PAGE 383

EXECUTION PHASE OF 9830 PLOTTER BLOCK

| | | | | | |
|-------|-------|--------|-------|---------|--------------------------------------|
| 1299 | 23071 | 067057 | JMP | WREXC | |
| 1300* | | | | | |
| 1301 | 23072 | 075753 | WRITN | SBP | *-1 |
| 1302 | 23073 | 074742 | SBR | 16 | CLEAR FORMAT NESTING FLAG |
| 1303 | 23074 | 035455 | STB | FORMAT | |
| 1304 | 23075 | 067026 | JMP | ELAB6 | |
| 1305* | | | | | |
| 1306 | 23076 | 000103 | BLNKA | DEF | .32+.32+1 |
| 1307 | 23077 | 163540 | WRITX | JSM | FSPR,I |
| 1308 | 23100 | 020012 | LDA | .1 | |
| 1309 | 23101 | 027076 | LDB | BLNKA | |
| 1310 | 23102 | 063203 | WRITQ | JSM | OUTCH |
| 1311 | 23103 | 067057 | JMP | WREXC | |
| 1312* | | | | | |
| 1313* | | | | | |
| 1314 | 23104 | 025466 | WRITF | LDB | LOCL1 STACK POINTER |
| 1315 | 23105 | 060563 | JSM | SLWST | |
| 1316 | 23106 | 163510 | JSM | CKEOL,I | |
| 1317 | 23107 | 074742 | SBR | 16 | |
| 1318 | 23110 | 067173 | JMP | WREND-2 | |
| 1319 | 23111 | 163564 | JSM | QTCHK,I | QUOTE NEXT IN WRITE? |
| 1320 | 23112 | 067115 | JMP | WRIT5 | P+1 NO |
| 1321 | 23113 | 063203 | JSM | OUTCH | P+2, YES (A) = LENGTH, (B) = POINTER |
| 1322 | 23114 | 067106 | JMP | WRITF+2 | |
| 1323* | | | | | |
| 1324 | 23115 | 163526 | WRIT5 | JSM | PFORM,I |
| 1325 | 23116 | 061220 | JSM | FETCH+1 | |
| 1326 | 23117 | 060567 | JSM | UNSTK | |
| 1327 | 23120 | 031466 | STA | LOCL1 | RECALL FORMAT STATEMENT POINTER |
| 1328 | 23121 | 060567 | JSM | UNSTK | |
| 1329 | 23122 | 031705 | STA | FLOAT | RECALL E, F OR B FLAG |
| 1330 | 23123 | 045466 | ISZ | LOCL1 | |
| 1331 | 23124 | 121466 | LDA | LOCL1,I | |
| 1332 | 23125 | 045466 | ISZ | LOCL1 | |
| 1333 | 23126 | 045466 | ISZ | LOCL1 | |
| 1334 | 23127 | 025456 | WR'TB | LDB | FORMS |
| 1335 | 23130 | 074073 | SBP | *+1,C | |
| 1336 | 23131 | 035456 | STB | FORMS | SET NUMERIC SPEC FOUND FLAG |
| 1337 | 23132 | 125466 | LDB | LOCL1,I | EGT DECIMAL SPEC |
| 1338 | 23133 | 163540 | JSM | FSPR,I | UPDATE FORMAT POINTERS |
| 1339 | 23134 | 031704 | STA | WIDTH | |
| 1340* | | | | | |
| 1341 | 23135 | 035703 | WRIT7 | STB | PLACE |
| 1342 | 23136 | 021412 | LDA | LSTPT | START TEMP BUFFER |
| 1343 | 23137 | 070070 | SIA | *+1 | |
| 1344 | 23140 | 070744 | SAL | 1 | |
| 1345 | 23141 | 031400 | STA | BADDR | |
| 1346 | 23142 | 031706 | STA | SBPTR | |
| 1347 | 23143 | 021413 | LDA | HSTPT | |
| 1348 | 23144 | 070744 | SAL | 1 | |
| 1349 | 23145 | 031623 | STA | PEND | |
| 1350 | 23146 | 021416 | LDA | MODE | SET END OF BUFFER AT HIGH STACK |
| 1351 | 23147 | 031737 | STA | T2 | |
| 1352 | 23150 | 070742 | SAR | 16 | |
| 1353 | 23151 | 031416 | STA | MODE | SET FOR PROGRAM FORMAT |
| 1354 | 23152 | 024215 | LDB | TMPOP | * |

498

PAGE 384

EXECUTION PHASE OF 9830 PLOTTER BLOCK

| | | | | | | |
|-------|-------|--------|-------|-----|---------|---|
| 1355 | 23153 | 060406 | | JSM | XFARI+1 | * OUTPUT NUMBER TO BUFFER |
| 1356 | 23154 | 160234 | | JSM | NMOTA,I | * |
| 1357 | 23155 | 021737 | | LDA | T2 | |
| 1358 | 23156 | 031416 | | STA | MODE | RESTORE THE MODE FLAG |
| 1359 | 23157 | 021400 | | LDA | BADDR | |
| 1360 | 23160 | 011623 | | CPA | PEND | OVERFLOW? |
| 1361 | 23161 | 064561 | | JMP | MER8 | |
| 1362 | 23162 | 025706 | | LDB | SBPTR | |
| 1363 | 23163 | 060374 | | JSM | WBUFF | |
| 1364 | 23164 | 063203 | | JSM | OUTCH | |
| 1365 | 23165 | 025625 | | LDB | FIRST | |
| 1366 | 23166 | 035400 | | STB | BADDR | RESET BADDR |
| 1367* | | | | | | |
| 1368 | 23167 | 163510 | WRIT6 | JSM | CKEOL,I | IS THIS THE END OF STATEMENT? |
| 1369 | 23170 | 074742 | | SBR | 16 | P+1, YES CRLF NOT SURPPRESSED |
| 1370 | 23171 | 067175 | | JMP | WREND | P+2, YES |
| 1371 | 23172 | 067057 | | JMP | WREXC | P+3, NO RETURN FOR MORE |
| 1372* | | | | | | |
| 1373 | 23173 | 055412 | | DSZ | LSTPT | |
| 1374 | 23174 | 055412 | | DSZ | LSTPT | |
| 1375 | 23175 | 070742 | WREND | SAR | 16 | |
| 1376 | 23176 | 031455 | | STA | FORMAT | |
| 1377* | | | | | | |
| 1378 | 23177 | 074110 | ELAB5 | SZB | *+2 | |
| 1379 | 23200 | 066554 | | JMP | EPEN | |
| 1380 | 23201 | 063367 | | JSM | PCRLF | |
| 1381 | 23202 | 164172 | | JMP | XEC4A,I | |
| 1382* | | | | | | |
| 1383 | 23203 | 031741 | OUTCH | STA | XC+1 | |
| 1384 | 23204 | 035740 | | STB | XC | |
| 1385 | 23205 | 062711 | | JSM | GETLC | SET-UP TI(POINTER TO CRLF) IT MAY BE KILLED |
| 1386 | 23206 | 061053 | .OUTC | JSM | GTMP | |
| 1387 | 23207 | 063213 | | JSM | PLET-1 | |
| 1388 | 23210 | 055741 | | DSZ | XC+1 | |
| 1389 | 23211 | 067206 | | JMP | .OUTC | |
| 1390 | 23212 | 170402 | | RET | | |
| 1391* | | | | | | |
| 1392* | | | | | | |
| 1393* | | | | | | |
| 1394 | 23213 | 061363 | | JSM | GTCON+1 | CONVERT FROM LOWER CASE |
| 1395 | 23214 | 003620 | PLET | ADA | M137B | CHECK IF > 137B |
| 1396 | 23215 | 070153 | | SAP | *+3 | SKIP IF THE CODE WAS > 172B |
| 1397 | 23216 | 000054 | | ADA | .63 | (B77) WITHIN RANGE ADJUST SO THAT 40B = 0 |
| 1398 | 23217 | 070113 | | SAP | *+2 | |
| 1399 | 23220 | 170402 | | RET | | RETURN IF NOT A VALID CHARACTER |
| 1400 | 23221 | 155647 | | DSZ | T1,I | BUMP THE CHARACTER COUNT FOR CRLF |
| 1401 | 23222 | 074742 | | SBR | 16 | |
| 1402 | 23223 | 035736 | | STB | CPX | |
| 1403 | 23224 | 035737 | | STB | CPY | |
| 1404 | 23225 | 070744 | | SAL | 1 | MULTIPLY BY 2 |
| 1405 | 23226 | 003476 | | ADA | VTABE | ADD IN THE VECTOR TABLE POINTER |
| 1406 | 23227 | 001645 | | ADA | CP4 | ADD IN CURRENT PAGE |
| 1407 | 23230 | 024213 | | LDB | ARIA | |
| 1408 | 23231 | 170004 | | XFR | | TRANSFER THE TWO WORD VECTOR TO AR1+1 AR1+2 |
| 1409 | 23232 | 024132 | | LDB | M1 | |
| 1410 | 23233 | 035747 | | STB | AR1+3 | SET THE 9TH BLOCK TO GIVE END OF CHAR |

PAGE 385

EXECUTION PHASE OF 9830 PLOTTER BLOCK

| | | | | | | |
|-------|-------|--------|-------|-----|-------|---------------------------------------|
| 1411 | 23234 | 175400 | PL1 | DLS | | |
| 1412 | 23235 | 013530 | | CPA | B16 | WAS THIS THE CCODE FOR SPEACIAL TABLE |
| 1413 | 23236 | 067250 | | JMP | STABL | YES |
| 1414 | 23237 | 070046 | | RAR | 2 | |
| 1415 | 23240 | 070137 | | LDB | A | |
| 1416 | 23241 | 074642 | | SBR | 14 | |
| 1417 | 23242 | 050020 | | AND | 3 | |
| 1418 | 23243 | 010020 | | CPA | 3 | |
| 1419 | 23244 | 067264 | | JMP | PL2 | END OF CHARACTER DEFINITION |
| 1420 | 23245 | 063302 | | JSM | MPEN | |
| 1421 | 23246 | 062021 | | JSM | PENDN | |
| 1422 | 23247 | 067234 | | JMP | PL1 | |
| 1423* | | | | | | |
| 1424* | | | | | | |
| 1425* | | | | | | |
| 1426 | 23250 | 021745 | STABL | LDA | ARI+1 | |
| 1427 | 23251 | 070142 | | SAR | 4 | |
| 1428 | 23252 | 050115 | | AND | B1777 | |
| 1429 | 23253 | 041645 | | IOR | CP4 | |
| 1430 | 23254 | 070744 | | SAL | 1 | |
| 1431 | 23255 | 031750 | | STA | YC | |
| 1432 | 23256 | 024057 | STAB2 | LDB | .1000 | |
| 1433 | 23257 | 061041 | | JSM | GETCH | |
| 1434 | 23260 | 031466 | | STA | LOCL1 | |
| 1435 | 23261 | 070306 | | RAR | 7 | |
| 1436 | 23262 | 070353 | | SAR | *+7 | |
| 1437 | 23263 | 070251 | | SLA | *+5 | |
| 1438* | | | | | | |
| 1439 | 23264 | 062015 | PL2 | JSM | PENUP | |
| 1440 | 23265 | 070004 | | SAL | 16 | |
| 1441 | 23266 | 024061 | | LDB | .10 | |
| 1442 | 23267 | 067310 | | JMP | MPEN2 | |
| 1443* | | | | | | |
| 1444 | 23270 | 062021 | | JSM | PENDN | |
| 1445 | 23271 | 070111 | | SLA | *+2 | |
| 1446 | 23272 | 062015 | | JSM | PENUP | |
| 1447 | 23273 | 021466 | | LDA | LOCL1 | |
| 1448 | 23274 | 070106 | | RAR | 3 | |
| 1449 | 23275 | 070137 | | LDB | A | |
| 1450 | 23276 | 074602 | | SBR | 13 | |
| 1451 | 23277 | 050024 | | AND | .7 | |
| 1452 | 23300 | 063310 | | JSM | MPEN2 | |
| 1453 | 23301 | 067256 | | JMP | STAB2 | |
| 1454* | | | | | | |
| 1455 | 23302 | 031466 | MPEN | STA | LOCL1 | |
| 1456 | 23303 | 070744 | | SAL | 1 | |
| 1457 | 23304 | 001466 | | ADA | LOCL1 | DELTA Y * 3 IN STANDARD MODE |
| 1458 | 23305 | 035466 | | STB | LOCL1 | |
| 1459 | 23306 | 074744 | | SBL | 1 | |
| 1460 | 23307 | 005466 | | ADB | LOCL1 | DELTA X * 3 IN STANDARD MODE |
| 1461* | | | | | | |
| 1462 | 23310 | 031466 | MPEN2 | STA | LOCL1 | |
| 1463 | 23311 | 021736 | | LDA | CPX | |
| 1464 | 23312 | 035736 | | STB | CPX | |
| 1465 | 23313 | 070076 | | TCA | | |
| 1466 | 23314 | 074017 | | ADA | B | |

PAGE 386

EXECUTION PHASE OF 9830 PLOTTER BLOCK

| | | | | | |
|-------|-------|--------|-------|-------------|----------------------------------|
| 1467 | 23315 | 031734 | | STA DX | DELTA X |
| 1468 | 23316 | 021466 | | LDA LOCL1 | |
| 1469 | 23317 | 025737 | | LDB CPY | |
| 1470 | 23320 | 031737 | | STA CPY | |
| 1471 | 23321 | 074076 | | TCB | |
| 1472 | 23322 | 070037 | | ADB A | |
| 1473 | 23323 | 035735 | MPEN3 | STB DY | DELTA Y |
| 1474 | 23324 | 121640 | | LDA YH,I | |
| 1475 | 23325 | 125641 | | LDB XH,I | |
| 1476 | 23326 | 063336 | | JSM MPY1 | $D.Y. = YH * D.Y. + XH * D.X.$ |
| 1477 | 23327 | 062117 | | JSM TYM | |
| 1478 | 23330 | 121643 | | LDA YW,I | |
| 1479 | 23331 | 125642 | | LDB XW,I | |
| 1480 | 23332 | 063336 | | JSM MPY1 | $D.X. = YW * D.Y. + XW * D.X.$ |
| 1481 | 23333 | 062125 | | JSM TXM | |
| 1482 | 23334 | 020107 | | LDA B1000 | |
| 1483 | 23335 | 066042 | | JMP MOVE-1 | |
| 1484* | | | | | |
| 1485 | 23336 | 031470 | MPY1 | STA L2T1 | |
| 1486 | 23337 | 021734 | | LDA DX | |
| 1487 | 23340 | 063355 | | JSM MPY2 | |
| 1488 | 23341 | 035471 | | STB L2T2 | |
| 1489 | 23342 | 025470 | | LDB L2T1 | |
| 1490 | 23343 | 021735 | | LDA DY | |
| 1491 | 23344 | 063355 | | JSM MPY2 | |
| 1492 | 23345 | 005471 | | ADB L2T2 | |
| 1493 | 23346 | 160227 | | JSM FXFLA,I | |
| 1494 | 23347 | 125647 | | LDB T1,I | |
| 1495 | 23350 | 014017 | | CPB .2 | WAS THIS VECTOR FLAGED AS A CRLF |
| 1496 | 23351 | 170402 | | RET | YES! |
| 1497 | 23352 | 020157 | MPY3 | LDA M256 | |
| 1498 | 23353 | 001754 | | ADA AR2 | |
| 1499 | 23354 | 066100 | | JMP STOR2 | |
| 1500* | | | | | |
| 1501 | 23355 | 070153 | MPY2 | SAP *+3 | |
| 1502 | 23356 | 070076 | | TCA | |
| 1503 | 23357 | 074076 | | TCB | |
| 1504 | 23360 | 035466 | | STB LOCL1 | |
| 1505 | 23361 | 074742 | | SBR 16 | |
| 1506 | 23362 | 070076 | | TCA | |
| 1507 | 23363 | 070170 | | SIA *+3 | |
| 1508 | 23364 | 005466 | | ADB LOCL1 | |
| 1509 | 23365 | 073770 | | RIA *-1 | |
| 1510 | 23366 | 170402 | | RET | |

501

488

1,444,141

488

PAGE 387 EXECUTION PHASE OF 9830 PLOTTER BLOCK

1512* LETTER AND LABEL ROUTINES

| | | | | | |
|-------|-------|--------|-------|-----|---------|
| 1513* | | | | | |
| 1514 | 23367 | 160170 | PCRLF | JSM | CLERA,I |
| 1515 | 23370 | 121647 | | LDA | T1,I |
| 1516 | 23371 | 031734 | | STA | DX |
| 1517 | 23372 | 020017 | | LDA | .2 |

SET THE FLAG SO MPEN3 DOESN'T
REDUCE BY 1/10TH

| | | | | | |
|------|-------|--------|-----|-----|-------|
| 1518 | 23373 | 131647 | | STA | T1,I |
| 1519 | 23374 | 024132 | | LDB | M1 |
| 1520 | 23375 | 063323 | | JSM | MPEN3 |
| 1521 | 23376 | 074742 | .PC | SBR | 16 |
| 1522 | 23377 | 135647 | | STB | T1,I |
| 1523 | 23400 | 170402 | | RET | |

1524* LETTER EXECUTION

| | | | | | |
|-------|-------|--------|------|-----|------|
| 1525* | | | | | |
| 1526* | | | | | |
| 1527 | 23401 | 062674 | EKEY | JSM | CDEF |

GET CURRENT PAGE, CHARACTER
LINKS AND CRLF

| | | | | | |
|------|-------|--------|-------|-----|---------|
| 1528 | 23402 | 074742 | | SBR | 16 |
| 1529 | 23403 | 035473 | | STB | CCNT |
| 1530 | 23404 | 020054 | | LDA | .63 |
| 1531 | 23405 | 061016 | | JSM | ONEXT |
| 1532 | 23406 | 163673 | .EKEY | JSM | KEYIN,I |
| 1533 | 23407 | 024141 | | LDB | M10 |
| 1534 | 23410 | 070346 | | RAR | 8 |
| 1535 | 23411 | 070133 | | SAP | *+2,C |
| 1536 | 23412 | 024132 | | LDB | M1 |
| 1537 | 23413 | 070346 | | RAR | 8 |
| 1538 | 23414 | 010034 | | CPA | .21 |
| 1539 | 23415 | 067446 | | JMP | .EK2 |
| 1540 | 23416 | 013532 | | CPA | B24 |
| 1541 | 23417 | 067447 | | JMP | .EK1 |
| 1542 | 23420 | 010036 | | CPA | B27 |
| 1543 | 23421 | 067453 | | JMP | .EK4 |
| 1544 | 23422 | 010035 | | CPA | B26 |
| 1545 | 23423 | 067454 | | JMP | .EK3 |
| 1546 | 23424 | 010104 | | CPA | B177 |
| 1547 | 23425 | 067442 | | JMP | .STOP |
| 1548 | 23426 | 010027 | | CPA | .11 |
| 1549 | 23427 | 067457 | | JMP | .RCRL |
| 1550 | 23430 | 013534 | | CPA | B157 |
| 1551 | 23431 | 020071 | | LDA | E |
| 1552 | 23432 | 074311 | | SLB | *+6 |
| 1553 | 23433 | 013055 | | CPA | B153 |
| 1554 | 23434 | 020101 | | LDA | B100 |
| 1555 | 23435 | 127506 | | LDB | VALUP,I |
| 1556 | 23436 | 163524 | | JSM | TBSRH,I |
| 1557 | 23437 | 067440 | | JMP | *+1 |
| 1558 | 23440 | 063214 | | JSM | PLET |
| 1559 | 23441 | 067406 | | JMP | .EKEY |

UP ARROW (25B)
DOWN ARROW (24B)
RIGHT ARROW (27B)
LEFT ARROW (26B)
STOP KEY?
YES

| | | | | | |
|-------|-------|--------|-------|-----|---------|
| 1560* | | | | | |
| 1561 | 23442 | 074742 | .STOP | SBR | 16 |
| 1562 | 23443 | 005442 | | STB | STPFL |
| 1563 | 23444 | 160170 | | JSM | CLERA,I |
| 1564 | 23445 | 164172 | | JMP | XEC4A,I |
| 1565* | | | | | |
| 1566 | 23446 | 074076 | .EK2 | TCB | |
| 1567 | 23447 | 070742 | .EK1 | SAR | 16 |

UP ARROW
ON ARROW

PAGE 388

EXECUTION PHASE OF 9830 PLOTTER BLOCK

| | | | | | | |
|-------|-------|--------|----------------|-----|---------|-------------------|
| 1568 | 23450 | 031734 | | STA | DX | |
| 1569 | 23451 | 063323 | | JSM | MPEN3 | |
| 1570 | 23452 | 067406 | | JMP | .EKEY | |
| 1571 | 23453 | 074076 | .EK4 | TCB | | RIGHT ARROW |
| 1572 | 23454 | 035734 | .EK3 | STB | DX | LEFT ARROW |
| 1573 | 23455 | 074742 | | SBR | 16 | |
| 1574 | 23456 | 067451 | | JMP | .EK1+2 | |
| 1575 | 23457 | 063367 | .RCRL | JSM | PCRLF | |
| 1576 | 23460 | 067402 | | JMP | EKEY+1 | |
| 1577* | | | | | | |
| 1578* | | | CHARACTER PLOT | | | |
| 1579* | | | | | | |
| 1580 | 23461 | 062674 | CPLT | JSM | CDEF | |
| 1581 | 23462 | 062263 | | JSM | FORM+2 | |
| 1582 | 23463 | 063471 | | JSM | .CPLT | |
| 1583 | 23464 | 035735 | | STB | DY | |
| 1584 | 23465 | 063471 | | JSM | .CPLT | |
| 1585 | 23466 | 035734 | | STB | DX | |
| 1586 | 23467 | 063324 | | JSM | MPEN3+1 | |
| 1587 | 23470 | 164172 | | JMP | XEC4A,I | |
| 1588* | | | | | | |
| 1589 | 23471 | 061220 | .CPLT | JSM | FETCH+1 | |
| 1590 | 23472 | 020013 | | LDA | C7 | MULT RESULT BY 10 |
| 1591 | 23473 | 063353 | | JSM | .MPY3+1 | |
| 1592 | 23474 | 066663 | | JMP | ROT1+2 | |

PAGE 389

LETTER AND NUMBER VICTOR TABLE

1594*
 1595* LETTER VECTOR TABLE
 1596*
 1597*
 1598 23475 167715 JMP BLKN,I SPACE 40B
 1599 23476 001474 VTABE DEF *-2-N9/6
 1600 23477 167713 JMP EXCM,I !
 1601 23500 001650 XMIN DEF XN
 1602 23501 167746 JMP QUOTE,I " 42B
 1603 23502 001660 YMIN DEF YN
 1604 23503 167716 JMP NUMBR,I #
 1605 23504 001664 YMAX DEF YX
 1606 23505 167734 JMP DOLLR,I \$
 1607 23506 002412 VALUP DEF 2412B ABS ADDRESS
 1608 23507 167701 JMP PERCT,I
 1609 23510 007227 CKEOL DEF 7227B ABS ADDRESS
 1610 23511 167723 JMP AMP,I &
 1611 23512 176000 EXPM4 OCT 176000
 1612 23513 112700 OCT 112700
 1613 23514 023417 .9999 DEC 9999
 1614 23515 167740 JMP LPARN,I S.T. POINTER FOR (
 1615 23516 012025 FMTOP OCT 12025
 1616 23517 167743 JMP RPARN,I S.T. POINTER FOR)
 1617 23520 177736 M34 DEC -34
 1618 23521 167674 JMP MULT,I *
 1619 23522 001674 TEMY DEF Z2
 1620 23523 167676 JMP PLUS,I
 1621 23524 003072 TBSRH DEF 3072B
 1622 23525 002700 DEC 1472
 1623 23526 006773 PFORM DEF 6773B ABS ADDRESS
 1624 23527 043300 DEC 18112
 1625 23530 000016 B16 OCT 16
 1626 23531 016000 DEC 7168
 1627 23532 000024 B24 OCT 24
 1628 23533 005300 DEC 2752 /
 1629 23534 000157 B157 OCT 157
 1630 23535 004242 OCT 4242,5300 0
 1631 23536 005300
 1631 23537 014700 DEC 6592 1
 1632 23540 007356 FSPR DEF 7356B ABS ADDRESS
 1633 23541 105145 OCT 105145,1300 2
 1634 23542 001300
 1634 23543 105145 DEC -30107,25100 3
 1635 23544 061014
 1635 23545 102152 DEC -31638,11264 4
 1636 23546 026000
 1636 23547 124105 OCT 124105,20300 5
 1637 23550 020300
 1637 23551 124106 DEC -22458,8268 6
 1638 23552 020114
 1638 23553 105034 OCT 105034 7
 1639 23554 177741 M31 DEC -31
 1640 23555 044246 DEC 18598,16422 8
 1641 23556 040046
 1641 23557 001250 DEC 680,18112 9
 23560 043300

SOH

PAGE 390

LETTER AND NUMBER VECTOR TABLE

| | | | | | |
|------|-------|--------|-------|-------------------|--------------------|
| 1642 | 23561 | 167710 | | JMP COLON,I | : |
| 1643 | 23562 | 004303 | RITE | DEF 4303B | RITE+1 ABS ADDRESS |
| 1644 | 23563 | 167707 | | JMP SCOLN,I | : |
| 1645 | 23564 | 007240 | QTCHK | DEF 7240B | ABS ADDRESS |
| 1646 | 23565 | 112034 | | DEC -27620 | < |
| 1647 | 23566 | 012761 | SIN | DEF 12761B | ABS ADDRESS |
| 1648 | 23567 | 167720 | | JMP EQUAL,I | = |
| 1649 | 23570 | 013007 | COS | DEF 13007B | ABS ADDRESS |
| 1650 | 23571 | 113034 | | DEC -27108 | > |
| 1651 | 23572 | 007402 | PLOTE | DEF 7402B | ABS ADDRESS |
| 1652 | 23573 | 167727 | | JMP QUES,I | ? |
| 1653 | 23574 | 006656 | PRINN | DEF 6656B | |
| 1654 | 23575 | 004242 | | OCT 4242,12554 | @ 100B |
| | 23576 | 012554 | | | |
| 1655 | 23577 | 004242 | | DEC 2210,25792 | A |
| | 23600 | 062300 | | | |
| 1656 | 23601 | 004245 | | DEC 2213,17952 | B |
| | 23602 | 043040 | | | |
| 1657 | 23603 | 124501 | | DEC -22207,11264 | C |
| | 23604 | 026000 | | | |
| 1658 | 23605 | 004226 | | DEC 2198,4288 | D |
| | 23606 | 010300 | | | |
| 1659 | 23607 | 124105 | | DEC -22459,16428 | E |
| | 23610 | 040054 | | | |
| 1660 | 23611 | 124105 | | DEC -22459,16576 | F |
| | 23612 | 040300 | | | |
| 1661 | 23613 | 124002 | | DEC -22526,26048 | G |
| | 23614 | 062700 | | | |
| 1662 | 23615 | 004106 | | DEC 2118,23872 | H |
| | 23616 | 121300 | | | |
| 1663 | 23617 | 014700 | | DEC 6592 | I |
| 1664 | 23620 | 177641 | M137B | OCT -137 | |
| 1665 | 23621 | 121004 | | DEC -24060,-16384 | J |
| | 23622 | 140000 | | | |
| 1666 | 23623 | 004112 | | DEC 2122,17088 | K |
| | 23624 | 041300 | | | |
| 1667 | 23625 | 100054 | | DEC -32724 | L |
| 1668 | 23626 | 000000 | | BSS 1 | CHECK SUM WORD |
| 1669 | 23627 | 004132 | | DEC 2138,11264 | M |
| | 23630 | 026000 | | | |
| 1670 | 23631 | 004052 | | DEC 2090,-16384 | N |
| | 23632 | 140000 | | | |
| 1671 | 23633 | 004242 | | DEC 2210,3072 | O |
| | 23634 | 006000 | | | |
| 1672 | 23635 | 004246 | | DEC 2214,19456 | P |
| | 23636 | 046000 | | | |
| 1673 | 23637 | 020212 | | DEC 8330,9664 | Q |
| | 23640 | 022700 | | | |
| 1674 | 23641 | 004246 | | DEC 2214,17708 | R |
| | 23642 | 042454 | | | |
| 1675 | 23643 | 001144 | | DEC 612,-30016 | S |
| | 23644 | 105300 | | | |
| 1676 | 23645 | 105221 | | DEC -30063,-16384 | T |
| | 23646 | 140000 | | | |
| 1677 | 23647 | 100052 | | DEC -32726,-16384 | U |
| | 23650 | 140000 | | | |

PAGE 391

LETTER AND NUMBER VECTOR TABLE

| | | | | | |
|-------|-------|--------|---|--------------------------|-------------|
| 1678 | 23651 | 100654 | | DEC -32340 | V |
| 1679 | 23652 | 007303 | TYPEC | DEF 7303B | ABS ADDRESS |
| 1680 | 23653 | 100122 | | DEC -32686,-21504 | W |
| | 23654 | 126000 | | | |
| 1681 | 23655 | 005130 | | DEC 2648,11264 | X |
| | 23656 | 026000 | | | |
| 1682 | 23657 | 102645 | | DEC -31323,7168 | Y |
| | 23660 | 016000 | | | |
| 1683 | 23661 | 105002 | | DEC -30206,-16384 | Z |
| | 23662 | 140000 | | | |
| 1684 | 23663 | 010211 | | OCT 10211,140000 | 133B |
| | 23664 | 140000 | | | |
| 1685 | 23665 | 101300 | | OCT 101300 | |
| 1686 | 23666 | 005555 | MOVCK | DEF 5555B | |
| 1687 | 23667 | 011251 | | OCT 11251,140000 | 135 B |
| | 23670 | 140000 | | | |
| 1688 | 23671 | 014511 | | OCT 14511,66000 | 136B |
| | 23672 | 066000 | | | |
| 1689* | | | | | |
| 1690 | 23502 | | PPPOS | EQU YMIN | |
| 1691* | | | | | |
| 1692* | | | MORE ABS ADRESSESS | | |
| 1693* | | | | | |
| 1694 | 23673 | 002753 | KEYIN | DEF 2753B | |
| 1695* | | | | | |
| 1696* | | | SPECIAL SYMBOL VECTOR TABLE | | |
| 1697* | | | | | |
| 1698* | | | BIT 15 = 1 PENUP BEFORE MOVEMENT | | |
| 1699* | | | BIT 14 = 1 PEN DOWN BEFORE MOVEMENT | | |
| 1700* | | | BITS 13,12,11 = Y COOR. OF 1ST LOCATION | | |
| 1701* | | | BIT 10,9,8 = X COOR. OF 1ST LOCATION | | |
| 1702* | | | BIT 7 = 1 PENUP | | |
| 1703* | | | BIT 6 = PEN DOWN | | |
| 1704* | | | BIT 5,4,0 = Y OF 2ND | | |
| 1705* | | | BIT 2,1,0 = X OF 2ND | | |
| 1706* | | | | | |
| 1707* | | | PEN CONDITIONS | | |
| 1708* | | | | | |
| 1709* | | | PEN UP ON ENTRY | | |
| 1710* | | | PEN UP ON EXIT | | |
| 1711* | | | A 7 IN Y INDICATES END OF CHAR. | | |
| 1712* | | | | | |
| 1713* | | | THIS TABLE WORKS ON A 7X7 MATRIX (6-0) | | |
| 1714* | | | | | |
| 1715 | 23674 | 104555 | MULT | OCT 104555,124515 | |
| | 23675 | 124515 | | | |
| 1716 | 23676 | 101563 | PLUS | OCT 101563,114136,140000 | + |
| | 23677 | 114136 | | | |
| | 23700 | 140000 | | | |
| 1717 | 23701 | 073161 | PERCT | OCT 73161,24041,25061 | % |
| | 23702 | 024041 | | | |
| | 23703 | 025061 | | | |
| 1718 | 23704 | 112514 | | OCT 112514,2416,12700 | |
| | 23705 | 002416 | | | |
| | 23706 | 012700 | | | |
| 1719 | 23707 | 000523 | SCOLN | OCT 523 | ; 73B |

506

PAGE 392

LETTER AND NUMBER VECTOR TABLE

| | | | | | | |
|------|-------|--------|-------|-----|---------------------|--------|
| 1720 | 23710 | 011523 | COLON | OCT | 11523,125553,140000 | : 72B |
| | 23711 | 125553 | | | | |
| | 23712 | 140000 | | | | |
| 1721 | 23713 | 031523 | EXCM | OCT | 31523 | ! 41B |
| 1722 | 23714 | 101503 | | OCT | 101503 | |
| 1723 | 23715 | 140000 | BLKN | OCT | 140000 | |
| 1724 | 23716 | 131102 | NUMBR | OCT | 131102,102164 | # |
| | 23717 | 102164 | | | | |
| 1725 | 23720 | 110126 | EQUAL | OCT | 110126 | = |
| 1726 | 23721 | 123140 | | OCT | 123140 | |
| 1727 | 23722 | 140000 | | OCT | 140000 | |
| 1728 | 23723 | 003152 | AMP | OCT | 3152 | & |
| 1729 | 23724 | 031454 | | OCT | 31454 | |
| 1730 | 23725 | 010403 | | OCT | 10403 | |
| 1731 | 23726 | 012700 | | OCT | 12700 | |
| 1732 | 23727 | 024562 | QUES | OCT | 24562 | ? |
| 1733 | 23730 | 031454 | | OCT | 31454 | |
| 1734 | 23731 | 022033 | | OCT | 22033 | |
| 1735 | 23732 | 011603 | | OCT | 11603 | |
| 1736 | 23733 | 041700 | | OCT | 41700 | |
| 1737 | 23734 | 004515 | DOLLR | OCT | 4515 | \$ 44B |
| 1738 | 23735 | 024455 | | OCT | 24455 | |
| 1739 | 23736 | 131503 | | OCT | 131503 | |
| 1740 | 23737 | 160000 | SC16 | OCT | 160000 | |
| 1741 | 23740 | 001511 | LPARN | OCT | 1511 | |
| 1742 | 23741 | 014051 | | OCT | 14051 | |
| 1743 | 23742 | 031700 | | OCT | 31700 | |
| 1744 | 23743 | 001515 | RPARN | OCT | 1515 | |
| 1745 | 23744 | 017055 | | OCT | 17055 | |
| 1746 | 23745 | 031700 | | OCT | 31700 | " |
| 1747 | 23746 | 021162 | QUOTE | OCT | 21162 | |
| 1748 | 23747 | 132144 | | OCT | 132144 | |
| 1749 | 23750 | 140000 | | OCT | 140000 | |

1750*

1751*

1752 23760

ORG 23760B

1753*

1754*

STATEMENT EXECUTION JUMP TABLE

1755*

| | | | | |
|------|-------|--------|------|---------|
| 1756 | 23760 | 177421 | ABS | EKEY-* |
| 1757 | 23761 | 177500 | ABS | CPL0T-* |
| 1758 | 23762 | 176303 | ABS | ESCAL-* |
| 1759 | 23763 | 176527 | ABS | EYAXE-* |
| 1760 | 23764 | 176525 | ABS | EXAXE-* |
| 1761 | 23765 | 176476 | ABS | EPL0T-* |
| 1762 | 23766 | 176361 | ABS | EOFF-* |
| 1763 | 23767 | 176565 | ABS | EPEN-* |
| 1764 | 23770 | 176744 | ABS | ELAB-* |
| 1765 | 23771 | 176471 | ABS | EIPLT-* |
| 1766 | 23772 | 001670 | TEMX | DEF Z1 |

OP CODE 1 IS NOT USED, SO USE IT
FOR A CONSTANT

1767*

1768 23773 176636

ABS TBL-*

1769 23774 176635

ABS TBL-*

1770 23775 176634

ABS TBL-*

1771 23776 176154

ABS SNAME-*

1772 23777 010000

OCT 10000

CODE TO IDENTIFY THE OPTION
BLOCK

507

494

1,444,141

494

PAGE 393

CROSS-REFERENCE TABLE

| | | | | | | |
|-------|------|------|------|------|------|------|
| .1 | 0020 | 1308 | | | | |
| .10 | 0071 | 0072 | 1441 | | | |
| .100 | 0070 | | | | | |
| .1000 | 0069 | 1432 | | | | |
| .11 | 0035 | 0043 | 1548 | | | |
| .12 | 0036 | | | | | |
| .13 | 0039 | | | | | |
| .15 | 0048 | 0083 | | | | |
| .16 | 0049 | 0084 | 0871 | 0895 | 0928 | 0961 |
| .1E4 | 0068 | | | | | |
| .2 | 0027 | 0610 | 0838 | 0964 | 1495 | 1517 |
| .21 | 0050 | 1538 | | | | |
| .22 | 0051 | 0087 | | | | |
| .23 | 0052 | 0088 | | | | |
| .28 | 0053 | | | | | |
| .3 | 0028 | 0045 | 0614 | 1417 | 1418 | |
| .31 | 0054 | 0090 | 0652 | | | |
| .32 | 0055 | 1306 | 1306 | | | |
| .33 | 0056 | | | | | |
| .34 | 0057 | 0091 | | | | |
| .37 | 0058 | | | | | |
| .4 | 0029 | 0857 | 0876 | 0917 | 0970 | 1070 |
| .40 | 0059 | | | | | |
| .41 | 0060 | | | | | |
| .43 | 0061 | | | | | |
| .45 | 0062 | | | | | |
| .46 | 0063 | | | | | |
| .48 | 0064 | 1116 | | | | |
| .5 | 0030 | | | | | |
| .58 | 0065 | 0094 | | | | |
| .6 | 0031 | 0044 | | | | |
| .63 | 0066 | 1397 | 1530 | | | |
| .7 | 0032 | 1451 | | | | |
| .8 | 0033 | 0883 | 0945 | 0952 | 1072 | |
| .9 | 0034 | | | | | |
| .99 | 0074 | | | | | |
| .9999 | 1613 | 0663 | 1066 | | | |
| .AX4 | 1084 | 1055 | | | | |
| .BUFA | 0180 | | | | | |
| .CDEF | 1175 | 1158 | | | | |
| .CL1 | 1090 | 1101 | | | | |
| .CL2 | 1093 | 1089 | 1103 | | | |
| .CPLT | 1589 | 1582 | 1584 | | | |
| .E1 | 1032 | 1063 | | | | |
| .EK1 | 1567 | 1541 | 1574 | | | |
| .EK2 | 1566 | 1539 | | | | |
| .EK3 | 1572 | 1545 | | | | |
| .EK4 | 1571 | 1543 | | | | |
| .EKEY | 1532 | 1559 | 1570 | | | |
| .ELAB | 1211 | 1187 | | | | |
| .FADA | 0185 | 0706 | 1045 | | | |
| .FDVA | 0188 | 0878 | 1073 | 1138 | | |
| .FMPA | 0187 | 0882 | 0948 | 1141 | 1223 | 1242 |
| .FSBA | 0186 | 0874 | 0904 | 0953 | | 1251 |
| .FSR | 0190 | | | | | |

PAGE 394

CROSS-REFERENCE TABLE

[illegible]

509

496

1,444,141

496

PAGE 395

CROSS-REFERENCE TABLE

| | | | | | | | |
|--------|------|------|------|------|------|------|------|
| B2400 | 0110 | | | | | | |
| B26 | 0087 | 1544 | | | | | |
| B27 | 0088 | 1542 | | | | | |
| B3000 | 0111 | | | | | | |
| B36 | 0089 | | | | | | |
| B37 | 0090 | | | | | | |
| B377 | 0099 | 0638 | 0646 | 1179 | | | |
| B400 | 0100 | 0639 | 0643 | 0647 | | | |
| B4000 | 0112 | 0162 | | | | | |
| B42 | 0091 | | | | | | |
| B44 | 0092 | | | | | | |
| B52 | 0093 | | | | | | |
| B6K | 0113 | 0167 | 1147 | | | | |
| B72 | 0094 | | | | | | |
| B777 | 0101 | | | | | | |
| BADDR | 0291 | 1276 | 1280 | 1345 | 1359 | 1366 | |
| BADRP | 0204 | | | | | | |
| BEND | 0198 | 0199 | | | | | |
| BHSTP | 0444 | | | | | | |
| BLANK | 0388 | | | | | | |
| BLKN | 1723 | 1598 | | | | | |
| BLNKA | 1306 | 1309 | | | | | |
| BUSY | 0617 | 0619 | 0636 | 0640 | 0644 | 0649 | |
| C5 | 0015 | | | | | | |
| C6 | 0011 | 0114 | | | | | |
| C7 | 0022 | 0100 | 1590 | | | | |
| CCNT | 0538 | 1529 | | | | | |
| CDEF | 1154 | 1176 | 1199 | 1527 | 1580 | | |
| CHRSA | 0219 | | | | | | |
| CKEOL | 1609 | 1269 | 1286 | 1316 | 1368 | | |
| CLAR2 | 0492 | | | | | | |
| CLERA | 0160 | 1265 | 1514 | 1563 | | | |
| CLIMT | 1089 | 1027 | 1031 | 1082 | | | |
| CLINE | 0316 | | | | | | |
| CLPIA | 0217 | | | | | | |
| COLON | 1720 | 1642 | | | | | |
| COMCE | 0436 | 0791 | 0804 | 0806 | | | |
| COMCK | 0435 | 0759 | 0762 | 0777 | 0780 | 0798 | |
| CONST | 0484 | | | | | | |
| COS | 1649 | 1230 | | | | | |
| CP2 | 0540 | 0927 | 0947 | 0951 | 0962 | | |
| CP3 | 0517 | 0938 | 0984 | 1016 | | | |
| CP4 | 0518 | 0877 | 0971 | 1134 | 1222 | 1406 | 1429 |
| CPLAC | 0331 | | | | | | |
| CPLLOT | 1580 | 1757 | | | | | |
| CPT | 0541 | 0866 | 0879 | 0884 | 0899 | 0905 | |
| CPX | 0535 | 0542 | 1402 | 1463 | 1464 | | |
| CPY | 0536 | 0543 | 1403 | 1469 | 1470 | | |
| CRCKA | 0228 | | | | | | |
| CRLF | 0501 | | | | | | |
| CRLF1 | 0502 | | | | | | |
| CRTN | 0037 | 0038 | 0039 | | | | |
| CTYPE | 0330 | | | | | | |
| CURPG | 1132 | 0819 | 0986 | 1015 | 1148 | 1154 | 1191 |
| DCOMA | 0210 | | | | | | |
| DEFLT | 1255 | 1203 | | | | | |

CROSS-REFERENCE TABLE

[illegible]

PAGE 397

CROSS-REFERENCE TABLE

| | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|--|
| FIRST | 0367 | 1274 | 1278 | 1281 | 1365 | | | | |
| FIX | 0505 | | | | | | | | |
| FLGBT | 0073 | | | | | | | | |
| FLOAT | 0386 | 1329 | | | | | | | |
| FLTFA | 0193 | | | | | | | | |
| FLTRA | 0192 | 0657 | 0992 | 1090 | 1142 | | | | |
| FLTSA | 0195 | | | | | | | | |
| FMTOP | 1615 | 1193 | | | | | | | |
| FN | 0148 | | | | | | | | |
| FND | 0462 | | | | | | | | |
| FND1A | 0224 | | | | | | | | |
| FNDAA | 0223 | | | | | | | | |
| FNDPS | 0476 | | | | | | | | |
| FOPBS | 0206 | | | | | | | | |
| FORM | 0810 | 0820 | 0891 | 0987 | 1025 | 1205 | 1581 | | |
| FORM2 | 0805 | 0790 | 0792 | 0797 | | | | | |
| FORM3 | 0803 | 0779 | | | | | | | |
| FORMA | 0172 | 0810 | 0811 | 0812 | 0813 | 1017 | 1021 | | |
| FORME | 0336 | | | | | | | | |
| FORMS | 0335 | 1334 | 1336 | | | | | | |
| FORMT | 0334 | 1303 | 1376 | | | | | | |
| FPAGE | 0451 | | | | | | | | |
| FROMA | 0122 | | | | | | | | |
| FSCA | 0173 | 0758 | 0761 | 0782 | 0800 | 0803 | 0805 | 0807 | |
| FSCE2 | 0441 | | | | | | | | |
| FSCE4 | 0446 | | | | | | | | |
| FSPR | 1632 | 1297 | 1307 | 1338 | | | | | |
| FTMP1 | 0337 | | | | | | | | |
| FTMP2 | 0338 | | | | | | | | |
| FWAM | 0157 | | | | | | | | |
| FWUP | 0296 | | | | | | | | |
| FXFLA | 0194 | 1053 | 1126 | 1493 | | | | | |
| GEI | 0675 | 0845 | 1159 | 1170 | 1172 | | | | |
| GETAD | 0461 | | | | | | | | |
| GETCH | 0471 | 1433 | | | | | | | |
| GETLC | 1167 | 1385 | | | | | | | |
| GETRW | 0672 | 0892 | 0996 | 1026 | 1065 | | | | |
| GETSK | 0475 | | | | | | | | |
| GFLAG | 0404 | | | | | | | | |
| GFRST | 0474 | | | | | | | | |
| GNEXT | 0473 | 0770 | 0776 | | | | | | |
| GPARM | 0424 | | | | | | | | |
| GSIGN | 0486 | | | | | | | | |
| GTCON | 0504 | 1394 | | | | | | | |
| GTMP | 0472 | 1386 | | | | | | | |
| GTOPP | 0466 | | | | | | | | |
| GTSTA | 0213 | | | | | | | | |
| HFLGI | 0321 | | | | | | | | |
| HSTPT | 0302 | 1347 | | | | | | | |
| I | 0067 | | | | | | | | |
| ICNT | 0422 | | | | | | | | |
| IMPY | 0487 | | | | | | | | |
| INF | 0077 | | | | | | | | |
| INITS | 0456 | | | | | | | | |
| INREA | 0226 | | | | | | | | |
| INTCK | 0459 | | | | | | | | |

PAGE 398 CROSS-REFERENCE TABLE

[illegible]

513

500

1,444,141

500

PAGE 399

CROSS-REFERENCE TABLE

| | | | | | |
|-------|------|------|------|------|------|
| M36 | 0140 | | | | |
| M4 | 0127 | | | | |
| M48 | 0141 | 1121 | | | |
| M5 | 0128 | | | | |
| M6 | 0129 | | | | |
| M64 | 0142 | | | | |
| M8 | 1091 | 0903 | | | |
| M80 | 0143 | | | | |
| M9 | 0130 | | | | |
| MASK1 | 0153 | | | | |
| MASK5 | 0154 | | | | |
| MAW | 0294 | | | | |
| MAXIN | 0400 | | | | |
| MAXSN | 0147 | 0661 | 1118 | | |
| MBIN1 | 0406 | | | | |
| MBIN2 | 0407 | | | | |
| MBOX1 | 0351 | | | | |
| MDIMA | 0208 | | | | |
| MER8 | 0445 | 1361 | | | |
| HINIT | 0455 | | | | |
| MODE | 0305 | 1261 | 1350 | 1353 | 1358 |
| MOVCK | 1686 | 0834 | | | |
| MOVE | 0633 | 1483 | | | |
| MPEN | 1455 | 1420 | | | |
| MPEN2 | 1462 | 1442 | 1452 | | |
| MPEN3 | 1473 | 1520 | 1569 | 1586 | |
| MPY1 | 1485 | 1476 | 1480 | | |
| MPY2 | 1501 | 1487 | 1491 | | |
| MSFLG | 0389 | | | | |
| MULT | 1715 | 1618 | | | |
| MVTOM | 0477 | | | | |
| N976 | 0588 | 1599 | | | |
| NDISP | 0382 | | | | |
| NDWRT | 0500 | | | | |
| NEXT1 | 0450 | | | | |
| NEXTA | 0467 | | | | |
| NEXTL | 0449 | | | | |
| NMOTA | 0200 | 1356 | | | |
| NOEOF | 0482 | | | | |
| NPRIN | 0383 | | | | |
| NPWRT | 0499 | | | | |
| NUM | 0396 | 0397 | | | |
| NUMBR | 1724 | 1604 | | | |
| NUMCA | 0211 | | | | |
| NUMOA | 0212 | | | | |
| NXTDT | 0304 | | | | |
| NXTST | 0405 | | | | |
| OBLNK | 0468 | | | | |
| ODGIT | 0496 | | | | |
| OF | 0493 | | | | |
| OFFS | 0792 | 0713 | 0718 | 0764 | |
| ONE | 0045 | 1185 | | | |
| ONEXT | 0469 | 1531 | | | |
| OPCHA | 0171 | | | | |
| OPDMK | 0079 | | | | |
| OPMSK | 0076 | 1201 | 1207 | | |

514

PAGE 400

CROSS-REFERENCE TABLE

| | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|--|
| OPTYP | 0397 | | | | | | | | |
| OTPTR | 0293 | | | | | | | | |
| OUTCH | 1383 | 1279 | 1310 | 1321 | 1364 | | | | |
| OUTCR | 0470 | | | | | | | | |
| OUTIA | 0179 | | | | | | | | |
| OVCHA | 0216 | | | | | | | | |
| OWHER | 0694 | 1127 | | | | | | | |
| PARMA | 0306 | | | | | | | | |
| PARMN | 0307 | | | | | | | | |
| PBPTO | 0313 | | | | | | | | |
| PBPTR | 0310 | | | | | | | | |
| PBUFF | 0311 | | | | | | | | |
| PBUFO | 0312 | | | | | | | | |
| PCRLF | 1514 | 1298 | 1380 | 1575 | | | | | |
| PEND | 0365 | 1349 | 1360 | | | | | | |
| PENDN | 0612 | 0606 | 0611 | 1085 | 1421 | 1444 | | | |
| PENUP | 0608 | 0671 | 0937 | 0980 | 1056 | 1155 | 1439 | 1446 | |
| PERCT | 1717 | 1608 | | | | | | | |
| PEXMA | 0165 | | | | | | | | |
| PFLAG | 0391 | | | | | | | | |
| PFORM | 1623 | 1324 | | | | | | | |
| PGIN0 | 0478 | | | | | | | | |
| PGINT | 0458 | | | | | | | | |
| PI/2 | 0044 | | | | | | | | |
| PL1 | 1411 | 1422 | | | | | | | |
| PLACE | 0384 | 1341 | | | | | | | |
| PLET | 1395 | 1387 | 1558 | | | | | | |
| PLOT | 0797 | 0717 | 0721 | | | | | | |
| PLOTE | 1651 | 1196 | | | | | | | |
| PLUS | 1716 | 1620 | | | | | | | |
| PMODE | 0329 | | | | | | | | |
| PNPAR | 0602 | 1002 | | | | | | | |
| PPFF | 0366 | | | | | | | | |
| PPPOS | 1690 | 1167 | | | | | | | |
| PRADD | 0308 | 0989 | 1019 | 1023 | | | | | |
| PRERR | 1149 | 1143 | 1295 | | | | | | |
| PRFLG | 0332 | 1268 | | | | | | | |
| PRINN | 1653 | 1272 | | | | | | | |
| PUN1 | 0943 | 0939 | | | | | | | |
| PUNIT | 0926 | 0919 | | | | | | | |
| PUNX | 0923 | 0911 | 1000 | | | | | | |
| PUNY | 0916 | 0912 | 0998 | | | | | | |
| PWHER | 0910 | 1075 | 1081 | 1088 | 1100 | | | | |
| QTCHA | 0230 | | | | | | | | |
| QTCHK | 1645 | 1319 | | | | | | | |
| QTOP | 0109 | 0161 | | | | | | | |
| QUES | 1732 | 1652 | | | | | | | |
| QUOTE | 1747 | 1602 | | | | | | | |
| RBOP | 0117 | | | | | | | | |
| RBUFF | 0360 | | | | | | | | |
| RCOUT | 0333 | | | | | | | | |
| RDYDA | 0166 | | | | | | | | |
| REED | 0161 | | | | | | | | |
| REEDL | 0339 | | | | | | | | |
| RES | 0417 | | | | | | | | |
| RESB1 | 0453 | | | | | | | | |

515

502

1,444,141

502

PAGE 401

CROSS-REFERENCE TABLE

| | | | | |
|--------|------|------|------|-----------|
| RESBP | 0454 | | | |
| RET2 | 0421 | | | |
| RETN2 | 0420 | | | |
| RETN3 | 0419 | | | |
| RITE | 1643 | 0783 | | |
| ROT1 | 1140 | 1233 | 1592 | |
| ROT2 | 1137 | 1237 | 1245 | |
| ROT4 | 1139 | 1252 | | |
| ROUND | 0495 | | | |
| RPARN | 1744 | 1616 | | |
| RPCK | 0439 | | | |
| RPCKE | 0440 | | | |
| RPOP | 0115 | 1202 | 1208 | |
| RSAVE | 0292 | | | |
| RSLTE | 0373 | | | |
| RSPTR | 0418 | 1112 | | |
| RSTAK | 0416 | | | |
| RSTKA | 0158 | | | |
| RESULT | 0426 | | | |
| RUNA | 0163 | | | |
| S | 0082 | | | |
| SALTA | 0207 | | | |
| SAVBP | 0452 | | | |
| SAVEA | 0325 | | | |
| SAVEB | 0326 | | | |
| SAXIS | 0758 | 0715 | 0716 | |
| SBADR | 0394 | | | |
| SBPTR | 0387 | 0784 | 1346 | 1362 |
| SBPUD | 0431 | 0768 | 0771 | 0775 |
| SBUFA | 0199 | | | |
| SC16 | 1740 | 0620 | 0622 | |
| SCAL | 0866 | 0858 | 0860 | |
| SCALE | 0790 | 0714 | | |
| SCOLN | 1719 | 1644 | | |
| SEC | 0320 | | | |
| SFLAG | 0393 | | | |
| SGNS | 0497 | | | |
| SIGN | 0399 | 0400 | | |
| SIN | 1647 | 1226 | | |
| SLCOD | 0328 | | | |
| SLEND | 0202 | | | |
| SLOCP | 1041 | | | |
| SLWST | 0447 | 1293 | 1315 | |
| SNAME | 0724 | 1771 | | |
| SNFLA | 0197 | | | |
| SPEN | 0754 | 0712 | 0719 | |
| SPLFA | 0196 | | | |
| SSBPT | 0395 | | | |
| SSYMA | 0170 | 0673 | 0823 | 1157 1168 |
| ST | 0377 | | | |
| STAB2 | 1432 | 1453 | | |
| STABL | 1426 | 1413 | | |
| STAR2 | 0490 | | | |
| START | 0162 | 0007 | | |
| STATU | 0622 | 0608 | 0612 | 0617 |
| STEAL | 0817 | 0672 | 0822 | 0841 |

CROSS-REFERENCE TABLE

| | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|
| STEXP | 0443 | | | | | | | | | | | |
| STMAX | 0494 | | | | | | | | | | | |
| STOR1 | 0428 | | | | | | | | | | | |
| STOR2 | 0664 | 1499 | | | | | | | | | | |
| STORE | 0427 | | | | | | | | | | | |
| STPFL | 0323 | 0651 | 1562 | | | | | | | | | |
| STROP | 0430 | | | | | | | | | | | |
| STSRH | 0442 | | | | | | | | | | | |
| SUMPA | 0227 | | | | | | | | | | | |
| SYE12 | 0485 | | | | | | | | | | | |
| SYE25 | 0460 | | | | | | | | | | | |
| SYMC1 | 0433 | 0772 | | | | | | | | | | |
| SYMC2 | 0432 | | | | | | | | | | | |
| SYMCK | 0434 | | | | | | | | | | | |
| SYMTP | 0298 | | | | | | | | | | | |
| SYTSA | 0209 | | | | | | | | | | | |
| T1 | 0520 | 0677 | 0885 | 0926 | 1400 | 1494 | 1515 | 1518 | 1522 | | | |
| T2 | 0543 | 0705 | 0707 | 0846 | 0849 | 0850 | 0930 | 0959 | 0972 | 1351 | 1357 | |
| T3 | 0519 | 0685 | 0694 | 0910 | 1013 | 1068 | 1078 | 1198 | 1255 | | | |
| T4 | 0542 | 0929 | 0932 | 0936 | 0940 | 0943 | | | | | | |
| TBL | 1112 | 1768 | 1769 | 1770 | | | | | | | | |
| TBSRH | 1621 | 1556 | | | | | | | | | | |
| TEMP | 0350 | 0351 | | | | | | | | | | |
| TEMP1 | 0352 | | | | | | | | | | | |
| TEMP2 | 0353 | | | | | | | | | | | |
| TEMP3 | 0354 | | | | | | | | | | | |
| TEMP4 | 0355 | | | | | | | | | | | |
| TEMP5 | 0539 | 0631 | 0648 | | | | | | | | | |
| TEMPS | 0349 | 0538 | 0539 | 0988 | 1018 | 1022 | 1192 | 1200 | 1206 | 1200 | | |
| TEMX | 1766 | 0702 | 1044 | 1244 | | | | | | | | |
| TEMXX | 0702 | 1033 | 1139 | 1243 | | | | | | | | |
| TEMY | 1619 | 1227 | 1236 | 1250 | | | | | | | | |
| TERM | 0364 | 1263 | 1266 | | | | | | | | | |
| TFLAG | 0318 | | | | | | | | | | | |
| TFLG1 | 0322 | | | | | | | | | | | |
| TICMK | 1099 | 1046 | | | | | | | | | | |
| TPMA | 0203 | | | | | | | | | | | |
| TPMOP | 0184 | 1354 | | | | | | | | | | |
| TR/W | 0885 | 0872 | 0880 | 0001 | 0906 | | | | | | | |
| TRANS | 0707 | 0941 | 0946 | 0949 | 0954 | | | | | | | |
| TREGE | 0372 | | | | | | | | | | | |
| TSRHA | 0168 | | | | | | | | | | | |
| TSTPT | 0300 | | | | | | | | | | | |
| TTY | 0503 | 0621 | | | | | | | | | | |
| TXM | 0696 | 0686 | 0875 | 1211 | 1481 | | | | | | | |
| TXX | 0699 | 0868 | 1030 | 1074 | 1099 | 1215 | | | | | | |
| TYM | 0688 | 0695 | 0870 | 1213 | 1231 | 1477 | | | | | | |
| TYPEC | 1679 | 1289 | | | | | | | | | | |
| TYPFL | 0078 | | | | | | | | | | | |
| TYX | 0691 | 1218 | 1224 | | | | | | | | | |
| UFLAG | 0392 | | | | | | | | | | | |
| UKPTR | 0309 | | | | | | | | | | | |
| UNITS | 0319 | | | | | | | | | | | |
| UNSTK | 0448 | 1326 | 1328 | | | | | | | | | |
| UPARA | 0191 | | | | | | | | | | | |
| VALUE | 0297 | 0826 | 0843 | | | | | | | | | |

CROSS-REFERENCE TABLE

[illegible]

PAGE 404

TERMINAL OPTION BLOCK

0503 LST
 0504 22000 ORG 22000PB
 0505*
 0506* TERMINAL OPTION BLOCK
 0507*
 0508 22000 000052 ID ABS XTRM-* COMMAND JUMP TABLE
 0509*
 0510 22001 052105 COMTB OCT 052105 TE
 0511 22002 051115 OCT 051115 RM
 0512 22003 170400 OCT 170400 (1) END ALLOWS TWO INTEGERS
 0513*
 0514*
 0515 22004 001757 ABS ERRX-* SYNTAX JUMP TABLE
 0516 22005 000000 NCP ** CHECKSUM WORD **
 0517*
 0518 22006 020302 SYSNT OCT 020302 (2) END
 0519*
 0520*
 0521 22007 000777 BN777 OCT 777
 0522*
 0523*
 0524 22010 005032 VALUP DEF 2415B+2415B
 0525 22011 002617 PPEFD DEF 2617B
 0526 22012 002753 KEYIN DEF 2753B
 0527 22013 003647 GETLN DEF 3647B
 0528 22014 006036 LISTS DEF 6036B
 0529 22015 006306 MCOUT DEF 6306B
 0530*
 0531*
 0532* SECONDARY ENTRY FROM MAIN SYSTEM THRU REED LINK
 0533*
 0534 22016 000017 RE1A DEF *+J-IO
 0535*
 0536 22017 023647 LDA ATIS
 0537 22020 027646 LDR ATI
 0538 22021 170004 XFR RESTORE CRITICAL PARAMETERS
 0539 22022 170004 XFR
 0540 22023 020062 LDA FLGBT
 0541 22024 031416 STA MODE RESTORE MODE FLAG
 0542*
 0543 22025 021433 LDA TFLAG
 0544 22026 010132 CPA M1 CHECK FOR COMPILE
 0545 22027 067104 JMP XCOM1
 0546 22030 070210 SZA *+4 CHECK FOR PTAPE
 0547 22031 162011 JSM PREED,I
 0548 22032 066031 JMP *-1 (NULL RECORD)
 0549 22033 066517 JMP EOLI
 0550*
 0551 22034 024212 LDB SLBF-
 0552 22035 074056 CMB
 0553 22036 005400 ADB BADDR
 0554 22037 004132 ADB M1 BADDR-, SLBF-2
 0555 22040 074113 SBP *+2 IS DISPLAY EMPTY?
 0556 22041 066134 JMP RE1 YES
 0557 22042 066147 JMP R17 NO, PRESERVE SYSTEM MESSAGE

519

506

1,444,141

506

PAGE 405

TERMINAL OPTION BLOCK

```

0559*
0560* "WHAT?" MESSAGE
0561*
0562 22043 062542 WHAT JSM MTIB SAVE MYSTERIOUS LINE
0563 22044 160170 JSM CLERA,I CLEAR THE DISPLAY
0564 22045 020022 LDA .5
0565 22046 027765 LDB WHAM
0566 22047 066145 JMP R10
0567*
0568*
0569 22050 000156 .110 DEC 110 CONSTANTS USED TO CHECK BAUD
                                     RATE
0570 22051 174702 M1598 DEC -1598
0571*
0572 22052 072110 XTRM RZA *+2 TEST FOR DEFAULT SELECT CODE (4)
0573 22053 020021 LDA .4
0574 22054 070204 SAL 12 SAVE SELECT CODE
0575 22055 031651 STA T1
0576 22056 014056 CPB .1E4 TEST FOR DEFAULT BAUD RATE (110)
0577 22157 026050 LDB .110
0578 22060 074117 LDA B
0579 22061 000134 ADA M3
0580 22062 070153 SAP *+3
0581 22063 160205 TERRI JSM AERRA,I ERROR, INVALID BAUD RATE
0582 22064 177662 DEC -78
0583 22065 000351 ADA M1598
0584 22066 071653 SAP *-3
0585 22067 160227 JSM FXFLA,I CALCULATE TIMING CONSTANT
0586 22070 020214 LDA AR2A
0587 22071 027650 LDB TFIA
0588 22072 170004 XFR
0589 22070 021717 LDA NUM
0590 22074 070442 SAR 10
0591 22075 070244 SAL 11
0592 22076 031654 STA T4 SAVE PAGE INFO
0593 22077 070002 SAR 1
0594 22100 031655 STA T5 TEMPORARY
0595 22101 003651 ADA TCON
0596 22102 027650 LDB TFIA
0597 22103 160221 JSM .FDVA,I
0598 22104 160225 JSM FLTRA,I
0599 22105 066105 JMP * OVERFLOW (SHOULD NEVER HAPPEN)
0600 22106 004025 ADB .8
0601 22107 035652 STB T2 SAVE TIMING CONSTANT
0602*
0603 22110 025655 LDB T5
0604 22111 006016 ADB REIA FORM NEW REED LINK
0605 22112 035462 STB REEDL
0606*
0607 22113 021651 LDA T1
0608 22114 070502 SAR 11 2 * SELECT CODE
0609 22115 003335 ADA INTAD + ADDRESS OF INTERRUPT TABLE
                                     - 1
0610 22116 025655 LDB T5
0611 22117 007336 ADB INTSA
0612 22120 070577 SIB A,I PUT INTERRUPT LINK ADDRESS IN
                                     TABLE

```

PAGE 406

TERMINAL OPTION BLOCK

| | | | | | |
|-------|-------|--------|-----|-------------|--------------------------------------|
| 0614 | 22121 | 070742 | | SAR 16 | |
| 0615 | 22122 | 031653 | | STA T3 | SET TO "EVEN" PARITY |
| 0616* | | | | | |
| 0617 | 22123 | 023641 | | LDA B4001 | TURN ON CARRIER |
| 0618 | 22124 | 041651 | | IOR T1 | |
| 0619 | 22125 | 172741 | | STF 1 | |
| 0620 | 22126 | 172141 | | OTA 1 | |
| 0621 | 22127 | 173601 | | STC 1,C | |
| 0622* | | | | | |
| 0623 | 22130 | 070742 | RE0 | SAR 16 | |
| 0624 | 22131 | 031430 | | STA LNINC | GET OUT OF AUTO |
| 0625 | 22132 | 031431 | | STA CLINE | INDICATE TOP OF PROGRAM |
| 0626 | 22133 | 031655 | | STA T5 | SET PRINT/STORE TO "PRINT" |
| 0627* | | | | | |
| 0628 | 22134 | 160170 | RE1 | JSM CLERA,I | CLEAR THE DISPLAY |
| 0629 | 22135 | 021430 | | LDA LNINC | |
| 0630 | 22136 | 070250 | | SZA *+5 | AUTO MODE? |
| 0631 | 22137 | 025427 | | LDB .LNUM | YES, OUTPUT LINE NUMBER |
| 0632 | 22140 | 160210 | | JSM OUTIA,I | |
| 0633 | 22141 | 045400 | | ISZ BADDR | FOLLOWED BY ONE BLANK |
| 0634 | 22142 | 066147 | | JMP R17 | |
| 0635* | | | | | |
| 0636 | 22143 | 020021 | | LDA .4 | NO, SUPPLY "TEXT" MESSAGE |
| 0637 | 22144 | 027223 | | LDB MSSG1 | |
| 0638 | 22145 | 005654 | R10 | ADB T4 | |
| 0639 | 22146 | 061335 | | JSM NPWRT | |
| 0640* | | | | | |
| 0641 | 22147 | 024236 | R17 | LDB SLEND | SET UP FOR KEYBOARD INPUT |
| 0642 | 22150 | 035623 | | STB PEND | FOR OUTCR |
| 0643 | 22151 | 070742 | | SAR 16 | |
| 0644 | 22152 | 031442 | | STA STPFL | CLEAR STOP FLAG |
| 0645 | 22153 | 031473 | | STA CCNT | MULTIPLE CHARACTERS VIA KEYIN |
| 0646 | 22154 | 031704 | | STA BKPTR | CURSOR POINTER |
| 0647 | 22155 | 031707 | | STA BLANK | DO NOT IGNORE BLANKS |
| 0648 | 22156 | 031657 | | STA T7 | SAY OK TO CLEAR DISPLAY ON INTERRUPT |
| 0649 | 22157 | 031660 | | STA T8 | CLEAR NO-LINE-NUMBERS FLAG |
| 0650* | | | | | |
| 0651 | 22160 | 162012 | | JSM KEYIN,I | CALL FOR FIRST KEY |
| 0652 | 22161 | 010105 | | CPA B377 | DUMP I/O BUFFER? |
| 0653 | 22162 | 066346 | | JMP DIOB | YES |
| 0654 | 22163 | 010074 | | CPA B22 | BACKSPACE? |
| 0655 | 22164 | 066420 | | JMP BKSP | YES |
| 0656 | 22165 | 031661 | | STA T9 | |
| 0657* | | | | | |
| 0658 | 22166 | 160170 | | JSM CLERA,I | CLEAR THE DISPLAY |
| 0659 | 22167 | 021430 | | LDA LNINC | |
| 0660 | 22170 | 070210 | | SZA *+4 | AUTO MODE? |
| 0661 | 22171 | 025427 | | LDB .LNUM | YES, OUTPUT LINE NUMBER |
| 0662 | 22172 | 160210 | | JSM OUTIA,I | |
| 0663 | 22173 | 045400 | | ISZ BADDR | FOLLOWED BY ONE BLANK |
| 0664 | 22174 | 021661 | | LDA T9 | |
| 0665 | 22175 | 066204 | | JMP R36 | |

521

508

1,444,141

508

PAGE 407

TERMINAL OPTION BLOCK

| | | | | | | |
|-------|-------|--------|-----|-----|---------|--------------------------------|
| 0667 | 22176 | 020212 | RE6 | LDA | .SLBF | |
| 0668 | 22177 | 011400 | | CPA | BADDR | DISPLAY EMPTY? |
| 0669 | 22200 | 066134 | | JMP | REI | |
| 0670 | 22201 | 162012 | R46 | JSM | KEYIN,I | CALL FOR NEXT KEY |
| 0671 | 22202 | 010105 | | CPA | B377 | DUMP I/O BUFFER? |
| 0672 | 22203 | 066346 | | JMP | DIOB | YES |
| 0673 | 22204 | 024104 | R36 | LDB | B177 | |
| 0674 | 22205 | 035663 | | STB | CTFLG | CLEAR ASCII-CNTRL FLAG |
| 0675 | 22206 | 074742 | | SBR | 16 | |
| 0676 | 22207 | 035662 | | STB | SHFLG | CLEAR ASCII-SHIFT FLAG |
| 0677 | 22210 | 074742 | R26 | SBR | 16 | |
| 0678 | 22211 | 070346 | | RAR | 8 | |
| 0679 | 22212 | 070133 | | SAP | *+2,C | TEST BIT 7 FOR SHIFT |
| 0680 | 22213 | 024132 | | LDB | M1 | |
| 0681 | 22214 | 035472 | | STB | TEMPS | SET SHIFT FLAG |
| 0682 | 22215 | 070346 | | RAR | 8 | |
| 0683 | 22216 | 031661 | | STA | T9 | INITIAL CODE WITHOUT SHIFT BIT |
| 0684* | | | | | | |
| 0685 | 22217 | 027656 | | LDB | EDITP | |
| 0686 | 22220 | 005654 | | ADB | T4 | |
| 0687 | 22221 | 160175 | | JSM | TSRHA,I | CODE IN EDIT TABLE? |
| 0688 | 22222 | 066245 | | JMP | R16 | NO |
| 0689 | 22223 | 002225 | | ADA | *+2 | |
| 0690 | 22224 | 070016 | | EXA | | YES, GO EXECUTE IMMEDIATELY |
| 0691* | | | | | | |
| 0692 | 22225 | 066226 | | JMP | *+1 | |
| 0693 | 22226 | 066564 | | JMP | OTPT 0 | EXECUTE |
| 0694 | 22227 | 066043 | | JMP | WHAT 1 | STEP |
| 0695 | 22230 | 066402 | | JMP | RECL 2 | RECALL |
| 0696 | 22231 | 066420 | | JMP | BKSP 3 | BACKSPACE |
| 0697 | 22232 | 066435 | | JMP | FORD 4 | FORWARDSpace |
| 0698 | 22233 | 066715 | | JMP | DNRW 5 | DOWNARROW |
| 0699 | 22234 | 066722 | | JMP | UPRW 6 | UPARROW |
| 0700 | 22235 | 066332 | | JMP | LEFT 7 | LEFTARROW |
| 0701 | 22236 | 066341 | | JMP | RITF 8 | RIGHTARROW |
| 0702 | 22237 | 066677 | | JMP | CLVR 9 | CLEAR VAR |
| 0703 | 22240 | 066450 | | JMP | INDL 10 | INSERT/DELETE |
| 0704 | 22241 | 066130 | | JMP | RE0 11 | CLEAR |
| 0705 | 22242 | 066662 | | JMP | REST 12 | ACC |
| 0706 | 22243 | 066622 | | JMP | PALL 13 | PRINT ALL |
| 0707 | 22244 | 066513 | | JMP | EOLL 14 | EOL |

PAGE 408

TERMINAL OPTION BLOCK

| | | | | | | |
|-------|-------|--------|-----|-----|---------|--------------------------------|
| 0709 | 22245 | 025472 | R16 | LDR | TEMPS | |
| 0710 | 22246 | 000141 | | ADA | M10 | DEFINED KEY? |
| 0711 | 22247 | 070253 | | SAP | RE2 | NO |
| 0712 | 22250 | 074150 | | SZB | *+3 | SHIFT? |
| 0713 | 22251 | 003644 | | ADA | .74 | YES |
| 0714 | 22252 | 031661 | | STA | T9 | |
| 0715 | 22253 | 067122 | | JMP | KEYPR | |
| 0716* | | | | | | |
| 0717 | 22254 | 003642 | RE2 | ADA | M22 | CODE < 40B? |
| 0718 | 22255 | 070612 | | SAM | R13 | YES |
| 0719 | 22256 | 000154 | | ADA | M64 | CODE < 140B? |
| 0720 | 22257 | 070353 | | SAP | RE3 | NO |
| 0721* | | | | | | |
| 0722 | 22260 | 074250 | | SZB | *+5 | SHIFT? |
| 0723 | 22261 | 000040 | | ADA | .31 | YES. LOWER CASE LETTER? |
| 0724 | 22262 | 070212 | | SAM | RE3 | NO |
| 0725 | 22263 | 003645 | | ADA | .97 | YES |
| 0726 | 22264 | 031661 | | STA | T9 | |
| 0727 | 22265 | 066314 | | JMP | RE5 | |
| 0728* | | | | | | |
| 0729 | 22266 | 021661 | RE3 | LDA | T9 | |
| 0730 | 22267 | 026010 | | LDB | VALUP | |
| 0731 | 22270 | 160175 | | JSM | TSRHA,I | CODE IN VALUE TABLE? |
| 0732 | 22271 | 066300 | R13 | JMP | RDD | NO |
| 0733 | 22272 | 000151 | | ADA | M32 | CODE < 40B? |
| 0734 | 22273 | 070113 | | SAP | *+2 | NO |
| 0735 | 22274 | 000101 | | ADA | B100 | YES |
| 0736 | 22275 | 000041 | | ADA | .32 | |
| 0737 | 22276 | 031661 | | STA | T9 | |
| 0738 | 22277 | 066314 | | JMP | RE5 | |
| 0739* | | | | | | |
| 0740 | 22300 | 021661 | RDD | LDA | T9 | |
| 0741 | 22301 | 027676 | | LDB | MULTP | |
| 0742 | 22302 | 005654 | | ADB | T4 | |
| 0743 | 22303 | 160175 | | JSM | TSRHA,I | CODE IN MULTI-CHARACTER TABLE? |
| 0744 | 22304 | 066043 | | JMP | WHAT | NO. QUESTION IT |
| 0745 | 22305 | 031661 | | STA | T9 | |
| 0746 | 22306 | 160170 | | JSM | CLERA,I | CLEAR THE DISPLAY |
| 0747 | 22307 | 021661 | | LDA | T9 | |
| 0748 | 22310 | 027714 | | LDB | MULTN | |
| 0749 | 22311 | 005654 | | ADB | T4 | |
| 0750 | 22312 | 162015 | | JSM | MCOUT,I | OUTPUT THE MNEMONIC |
| 0751 | 22313 | 066176 | | JMP | RE6 | |

PAGE 409

TERMINAL OPTION BLOCK

| | | | | | |
|-------|------------|--------|------|-----------|--|
| 0753 | 22314 | 021661 | RE5 | LDA T9 | ALL CONVERSION COMPLETE |
| 0754 | 22315 | 051662 | | AND SHFLG | EXCEPT FOR ASCII-SHIFT & ASCII-CNTRL |
| 0755 | 22316 | 070056 | | CMA | |
| 0756 | 22317 | 070137 | | LDB A | |
| 0757 | 22320 | 021661 | | LDA T9 | |
| 0758 | 22321 | 041662 | | IOR SHFLG | |
| 0759 | 22322 | 074257 | | AND B | |
| 0760 | 22323 | 051663 | | AND CTFLG | |
| 0761 | 22324 | 025704 | | LDB BKPTR | |
| 0762 | 22325 | 074210 | | SZB RE9 | CURSOR? |
| 0763 | 22326 | 027637 | | LDB BKPTA | YES |
| 0764 | 22327 | 061017 | | JSM OUTCR | REPLACE CHARACTER |
| 0765 | 22330 | 066440 | | JMP FW2 | SCROLL IF NECESSARY |
| 0766* | | | | | |
| 0767 | 22331 | 061016 | RE9 | JSM OMEXT | PUT CHARACTER AT END OF LINE |
| 0768* | | | | | AND SCROLL IF NECESSARY |
| 0769* | LEFTARROW | | | | |
| 0770* | | | | | |
| 0771 | 22332 | 021400 | LEFT | LDA BADDR | IS LAST CHARACTER AT RIGHT END OF DISPLAY? |
| 0772 | 22333 | 000151 | | ADA M32 | |
| 0773 | 22334 | 070076 | | TCA | |
| 0774 | 22335 | 001402 | | ADA OTPTR | |
| 0775 | 22336 | 070113 | | SAP *+2 | |
| 0776 | 22337 | 045432 | | ISZ OTPTR | NO, SCROLL |
| 0777 | 22340 | 066176 | JMP | RE6 | |
| 0778* | | | | | |
| 0779* | RIGHTARROW | | | | |
| 0780* | | | | | |
| 0781 | 22341 | 021402 | RITF | LDA OTPTR | IS FIRST CHARACTER AT LEFT END OF DISPLAY? |
| 0782 | 22342 | 010212 | | CPA SLRF | |
| 0783 | 22343 | 066176 | | JMP RE6 | |
| 0784 | 22344 | 055402 | | DSZ OTPTR | NO, SCROLL |
| 0785 | 22345 | 066176 | | JMP RE6 | |

PAGE 410

TERMINAL OPTION BLOCK

```

0787*
0788* DUMP I/O BUFFER FOR MODEM MESSAGE
0789*
0790 22346 021655 DIOB LDA IS
0791 22347 070110 SZA *+2
0792 22350 164203 JMP ACTSA,I
0793*
0794 22351 024211 LDB BUFA
0795 22352 035740 STB TMP
0796*
0797 22353 160261 JSM IOSTA,I IS PRINTER BUSY?
0798 22354 066147 JMP R17 (STOP KEY RETURN)
0799*
0800 22355 021740 DIO2 LDA TMP GET A CHARACTER
0801 22356 025740 LDB TMP
0802 22357 070002 SAR 1
0803 22360 070517 LDA A,I
0804 22361 074111 SLB *+2
0805 22362 070346 RAR 8
0806 22363 070342 SAR 8
0807 22364 010031 CPA EOL IS IT EOL?
0808 22365 066374 JMP DIO3 YES
0809 22366 041447 IOR SLCOD NO, OUTPUT IT
0810 22367 172741 STF 1
0811 22370 172141 OTA 1
0812 22371 173601 STC I,C
0813 22372 045740 ISZ TMP
0814 22373 066355 JMP DIO2
0815*
0816 22374 020061 DIO3 LDA LNFD
0817 22375 041447 ICR SLCOD
0818 22376 172741 STF 1
0819 22377 172141 OTA 1
0820 22400 173601 STC I,C
0821 22401 066147 JMP R17

```




PAGE 411

TERMINAL OPTION BLOCK

0825*
0826* RECALL
0827*
0828 22402 160170 RECL JSM CLERA,I CLEAR THE DISPLAY
0829 22403 024211 LDB BUFA
0830 22404 035740 STB TMP
0831 22405 061053 RECI JSM GTMP GET A CHARACTER
0832 22406 010031 CPA EOL
0833 22407 066176 JMP RE6 DONE
0834 22410 061016 JSM ONEXT OUTPUT TO DISPLAY
0835 22411 021400 LDA BADDR
0836 22412 000151 ADA M32
0837 22413 070076 TCA
0838 22414 001402 ADA OTPTR
0839 22415 070113 SAP *+2
0840 22416 045402 ISZ OTPTR
0841 22417 066405 JMP RECI
0842*
0843* BACKSPACE CURSOR
0844*
0845 22420 021714 BKSP LDA BKPTR
0846 22421 072310 RZA BK2 FIRST BACKSPACE?
0847 22422 021400 LDA BADDR YES
0848 22423 010212 CPA SLBF NULL RECORD?
0849 22424 066176 BK3 JMP RE6 YES
0850 22425 031704 STA BKPTR
0851 22426 066433 JMP BK1 PUT POINTER AT LAST CHARACTER
0852*
0853 22427 010212 BK2 CPA SLBF ALREADY AT FIRST CHARACTER
0854 22430 066176 JMP RE6 YES
0855 22431 011402 CPA OTPTR LEFTMOST ON SCREEN?
0856 22432 055402 DSZ OTPTR YES, SCROLL
0857 22433 055704 BK1 DSZ BKPTR BACKSPACE CURSOR
0858 22434 066176 JMP RE6
0859*
0860* FORWARDSPACE CURSOR
0861*
0862 22435 021704 FORD LDA BKPTR
0863 22436 071310 SZA BK3 IGNORE IF NO BACKSPACE HAS
OCCURRED
0864 22437 045704 ISZ BKPTR
0865 22440 021402 FW2 LDA OTPTR
0866 22441 000041 ADA 32
0867 22442 011704 CPA BKPTR ON RIGHT OF SCREEN?
0868 22443 011400 CPA BADDR YES, LINE FULL?
0869 22444 066176 JMP RE6
0870 22445 045402 ISZ OTPTR NO, SCROLL
0871 22446 066176 JMP RE6

PAGE 412

TERMINAL OPTION BLOCK

```

0873*
0874* INSERT/DELETE
0875*
0876 22447 066176 JMP RE6
0877 22450 025704 INDL LDB BKPTR CHECK CURSOR POSITION
0878 22451 075710 SZB *-2 ANY BACKSPACE YET?
0879 22452 021472 LDA TEMPS
0880 22453 070650 SZA INSR SHIFT FLAG SET?
0881*
0882* DELETE
0883*
0884 22454 055400 DSZ BADDR YES, MOVE END-OF-LINE POINTER
0885 22455 035740 STB TMP
0886 22456 015400 DLC1 CPB BADDR LAST CHARACTER?
0887 22457 066507 JMP INS2+1 YES, OUTPUT A BLANK
0888 22460 045740 ISZ TMP
0889 22461 061053 JSM GTMP GET CHARACTER
0890 22462 055740 DSZ TMP
0891 22463 055740 DSZ TMP
0892 22464 024111 LDB TMPA STORE SHIFTED LEFT ONE
0893 22465 061017 JSM OUTCR
0894 22466 025740 LDR TMP
0895 22467 066456 JMP DLC1
0896*
0897* INSERT
0898*
0899 22470 025400 INSR LDB BADDR
0900 22471 014236 CPB SLEND
0901 22472 066176 JMP RE6 LINE FULL
0902 22473 004132 ADB M1
0903 22474 035740 INSI STB TMP
0904 22475 061053 JSM GTMP GET CHARACTER
0905 22476 025740 LDB TMP
0906 22477 015704 CPB BKPTP AT CURSOR POSITION?
0907 22500 066506 JMP INS2 YES
0908 22501 024111 LDB TMPA NO, MOVE RIGHT ONE
0909 22502 061017 JSM OUTCR
0910 22503 025740 LDB TMP
0911 22504 004134 ADB M3 BACK UP TO NEXT CHARACTER TO LEFT
0912 22505 066474 JMP INSI
0913*
0914 22506 045400 INS2 ISZ BADDR UPDATE END-OF-LINE POINTER
0915 22507 020041 LDA .32
0916 22510 024111 LDB TMPA
0917 22511 061017 JSM OUTCR INSERT A BLANK
0918 22512 066176 JMP RE6

```

PAGE 413

TERMINAL OPTION BLOCK

0920*

0921* PROCESS STORE KEY

0922*

0923 22513 062542 EOLL

JSM MTIB

MOVE RECORD TO I/O BUFFER

0924 22514 060373

JSM ICNT

GET CHARACTER COUNT

0925 22515 010012

CPA .1

0926 22516 066134

JMP RE1

IGNORE NULL RECORD

0927*

0928 22517 062552 FOL1

JSM CRIT

SAVE CRITICAL PARAMETERS

0929 22520 061062

JSM GFRST

GET FIRST CHARACTER

0930 22521 066134

JMP RE1

(NULL RECORD RETURN)

0931 22522 060750

JSM DIGCK

DIGIT?

0932 22523 066620

JMP MAINS

NO, GIVE TO SYSTEM

0933*

0934 22524 060675

JSM INTCK-1

BUILD LINE #

0935 22525 035427

STB .LNUM

STORE LINE #

0936 22526 010031

CPA EOL

END OF LINE?

0937 22527 164204

JMP DELSA,I

YES, DELETE LINE

0938 22530 023640

LDA B5K

OPCODE 2 W/SPECIAL BIT

0939 22531 131706

STA SBPTR,I

0940 22532 045706

ISZ SBPTR

0941 22533 023777

LDA OPCOD

0942 22534 131706

STA SBPTR,I

0943 22535 055400

DSZ BADDR

0944 22536 020031

LDA EOL

0945 22537 024012

LDB .1

0946 22540 160253

JSM CHRSA,I

0947 22541 164203

JMP ACTSA,I

0948*

0949*

0950* SUBROUTINE TO MOVE RECORD TO I/O BUFFER

0951*

0952 22542 045623 MTIR

ISZ PEND

GUARANTEE ROOM FOR EOL

0953 22543 020031

LDA EOL

0954 22544 061016

JSM ONEXT

STORE EOL

0955 22545 024233

LDB BEND

0956 22546 035476

STB TEMP3

0957 22547 035477

STB TEMP4

0958 22550 004046

ADB .41

0959 22551 065122

JMP MVTOH+1

MOVE THE RECORD AND RETURN

0960*

0961* SUBROUTINE TO SAVE CRITICAL PARAMETERS (T1 THRU T7)

0962*

0963 22552 025450 CRIT

LDB PMODE

0964 22553 076210

RZB *+4

PRINT ALL?

0965 22554 024211

LDB .BUFA

YES

0966 22555 061337

JSM NDWRT

(A—REG HAS COUNT)

0967 22556 061344

JSM CRLF1

0968*

0969 22557 023646

LDA AT1

0970 22560 027647

LDB AT1S

0971 22561 170004

XFR

0972 22562 170004

XFR

0973 22563 170402

RET

528

PAGE 414

TERMINAL OPTION BLOCK

```

0975*
0976* PROCESS EXECUTE KEY
0977*
0978 22564 062542 OTPT JSM MTIB MOVE RECORD TO I/O BUFFER
0979 22565 060373 JSM ICNT GET CHARACTER COUNT
0980 22566 010012 CPA .1
0981 22567 066134 JMP RE1 IGNORE NULL RECORD
0982*
0983 22570 062552 JSM CRIT SAVE CRITICAL PARAMETERS
0984 22571 061062 JSM GFRST GET FIRST CHARACTER
0985 22572 000000 NOP (NULL RECORD RETURN)
0986 22573 027714 LDR MULTN
0987 22574 005654 ADB T4
0988 22575 160175 JSM TSRHA,I SEARCH MULTI-CHARACTER
                                TABLE
0989 22576 066620 JMP MAINS NOT FOUND, GIVE TO MAIN
                                SYSTEM
0990 22577 002601 ADA *+2
0991 22600 070016 EXA
0992*
0993 22601 066602 JMP *+1
0994 22602 067013 JMP XLIST 0 LIST
0995 22603 066620 JMP MAINS 1 CONT
0996 22604 066620 JMP MAINS 2 TRACE
0997 22605 066620 JMP MAINS 3 RUN
0998 22606 066710 JMP XFECH 4 FETCH
0999 22607 066620 JMP MAINS 5 LOAD
1000 22610 066620 JMP MAINS 6 STORE
1001 22611 066620 JMP MAINS 7 NORMAL
1002 22612 066620 JMP MAINS 8 FIX
1003 22613 066620 JMP MAINS 9 FLT
1004 22614 066620 JMP MAINS 10 SCRATCH
1005 22615 066620 JMP MAINS 11 AUTO
1006 22616 066701 JMP XEXIT 12 EXIT ("STOP" KEY)
1007*
1008 22617 067067 JMP XCOMP13 COMP
1009*
1010 22620 045416 MAINS ISZ MODE INDICATE CALC MODE
1011 22621 064357 JMP RETN2 RETURN TO MAIN SYSTEM

```

PAGE 415

TERMINAL OPTION BLOCK

| | | | | | |
|-------|---------------|--------|------|-----------------|-------------------|
| 1013* | | | | | |
| 1014* | PRINT ALL | | | | |
| 1015* | | | | | |
| 1016 | 22622 | 160170 | PALL | JSM CLERA,I | CLEAR THE DISPLAY |
| 1017 | 22623 | 021450 | | LDA PMODE | REVERSE THE FLAG |
| 1018 | 22624 | 070130 | | SIA *+2 | |
| 1019 | 22625 | 070742 | | SAR 16 | |
| 1020 | 22626 | 031450 | | STA PMODE | |
| 1021 | 22627 | 026634 | | LDB MON | |
| 1022 | 22630 | 070110 | | SZA *+2 | |
| 1023 | 22631 | 026637 | | LDB MOF | |
| 1024 | 22632 | 020020 | | LDA .3 | |
| 1025 | 22633 | 066145 | | JMP R10 | |
| 1026* | | | | | |
| 1027 | 22634 | 001472 | MON | DEF *+*+2-ID-ID | |
| 1028 | 22635 | 047516 | | ASC 2,ON | |
| | 22636 | 020040 | | | |
| 1029 | 22637 | 001500 | MOF | DEF *+*+2-ID-ID | |
| 1030 | 22640 | 047506 | | ASC 2,OFF | |
| | 22641 | 043040 | | | |
| 1031* | | | | | |
| 1032* | CHANGE PARITY | | | | |
| 1033* | | | | | |
| 1034 | 22642 | 160170 | EVOD | JSM CLERA,I | CLEAR THE DISPLAY |
| 1035 | 22643 | 021653 | | LDA T3 | REVERSE THE FLAG |
| 1036 | 22644 | 072113 | | SAP *+2,S | |
| 1037 | 22645 | 070073 | | SAP *+1,C | |
| 1038 | 22646 | 031653 | | STA T3 | |
| 1039 | 22647 | 026654 | | LDB MEV | |
| 1040 | 22650 | 070113 | | SAP *+2 | |
| 1041 | 22651 | 026657 | | LDB MOD | |
| 1042 | 22652 | 020021 | | LDA .4 | |
| 1043 | 22653 | 066145 | | JMP R10 | |
| 1044* | | | | | |
| 1045 | 22654 | 001532 | MEV | DEF *+*+2-ID-ID | |
| 1046 | 22655 | 042526 | | ASC 2,EVEN | |
| | 22656 | 042516 | | | |
| 1047 | 22657 | 001540 | MOD | DEF *+*+2-ID-ID | |
| 1048 | 22660 | 047504 | | ASC 2,ODD | |
| | 22661 | 042040 | | | |

PAGE 416

TERMINAL OPTION BLOCK

| | | | | | | |
|-------|-------------------------|--------|-------|-----|-------------|-------------------|
| 1050* | | | | | | |
| 1051* | ACC | | | | | |
| 1052* | | | | | | |
| 1053 | 22662 | 021472 | REST | LDA | TEMPS | CHECK SHIFT FLAG |
| 1054 | 22663 | 070210 | | SZA | *+4 | |
| 1055 | 22664 | 020101 | | LDA | B100 | USE "@" |
| 1056 | 22665 | 031661 | | STA | T9 | |
| 1057 | 22666 | 066314 | | JMP | RE5 | |
| 1058* | | | | | | |
| 1059 | 22667 | 020020 | | LDA | .3 | USE "RES" |
| 1060 | 22670 | 026674 | | LDB | RESM | |
| 1061 | 22671 | 005654 | | ADB | T4 | |
| 1062 | 22672 | 061335 | | JSM | NPWRT | |
| 1063 | 22673 | 066176 | | JMP | RE6 | |
| 1064* | | | | | | |
| 1065 | 22674 | 001572 | RESM | DEF | *+*+2-ID-ID | |
| 1066 | 22675 | 051105 | | ASC | 2,RES | |
| | 22676 | 051440 | | | | |
| 1067* | | | | | | |
| 1068* | CLEAR VAR | | | | | |
| 1069* | | | | | | |
| 1070 | 22677 | 020056 | CLVR | LDA | .1E4 | |
| 1071 | 22700 | 164171 | | JMP | RUNA,I | |
| 1072* | | | | | | |
| 1073* | EXIT FROM TERMINAL MODE | | | | | |
| 1074* | | | | | | |
| 1075 | 22701 | 021651 | XEXIT | LDA | T1 | TURN OFF CARRIER |
| 1076 | 22702 | 172741 | | STF | 1 | |
| 1077 | 22703 | 172141 | | OTA | 1 | |
| 1078 | 22704 | 173601 | | STC | 1,C | |
| 1079 | 22705 | 024116 | | LDB | QTOP | (=2000B) |
| 1080 | 22706 | 035462 | | STB | REEDL | RESTORE REED LINK |
| 1081 | 22707 | 164174 | | JMP | RDYDA,I | |

PAGE 417

TERMINAL OPTION BLOCK

```

1083*
1084*  FETCH
1085*
1086  22710  162013  XFECH  JSM  GETLN,I  GET PARAMETER
1087  22711  020012      LDA  .1
1088  22712  066713      JMP  +1      NOP
1089  22713  000132      ADA  M1
1090  22714  031431      STA  CLINF
1091*
1092*  DOWNARROW
1093*
1094  22715  025424  DNRW  LDB  PBUFF  CHECK FOR NULL PROGRAM
1095  22716  015423      CPB  PBPTR
1096  22717  066134      JMP  REI
1097  22720  062731      JSM  DNRWS
1098  22721  066726      JMP  UPR1
1099*
1100*  UPARROW
1101*
1102  22722  025424  UPRW  LDB  PBUFF  CHECK FOR NULL PROGRAM
1103  22723  015423      CPB  PBPTR
1104  22724  066134      JMP  REI
1105  22725  062737      JSM  UPRWS
1106  22726  070742  UPR1  SAR  16
1107  22727  031430      STA  LNINC  GET OUT OF AUTO.
1108  22730  066147      JMP  R17
1109*
1110*  DOWNARROW SUBROUTINE
1111*
1112  22731  021431  DNRWS  LDA  CLINE
1113  22732  070070      SIA  *+1
1114  22733  061075      JSM  FNDPS  FIND LINE
1115  22734  066747      JMP  U3      PAST END OF PROGRAM
1116  22735  066753      JMP  U5      BETWEEN LINES
1117  22736  066753      JMP  U5
1118*
1119*  UPARROW SUBROUTINE
1120*
1121  22737  021431  UPRWS  LDA  CLINE
1122  22740  072110      RZA  *+2  CLINE=0?
1123  22741  020056      LDA  .1E4  YES, FIND LAST LINE OF PROGRAM
1124  22742  061075      JSM  FNDPS  FIND LINE
1125  22743  066747      JMP  U3      PAST END OF PROGRAM
1126  22744  066745      JMP  *+1  BETWEEN LINES
1127  22745  015424      CPB  PBUFF  FIRST LINE?
1128  22746  066753      JMP  U5      YES
1129*
1130  22747  021720  U3      LDA  LASTN  GET PREVIOUS LINE
1131  22750  060742      JSM  FND
1132  22751  070742      SAR  16
1133  22752  031656      STA  T6
1134*
1135  22753  035661  U5      STB  T9      SAVE ADDRESS
1136  22754  035472      STB  TEMPS

```

PAGE 418

TERMINAL OPTION BLOCK

```

1138*
1139* SUBROUTINE TO OUTPUT A LINE TO DISPLAY BUFFER
1140*
1141* LOCATION IS IN T3
1142*
1143 22755 160170 LNOUT JSM CLERA,I CLEAR THE DISPLAY
1144 22756 125661 LDB T9,I GET LINE #
1145 22757 045661 ISZ T9
1146 22760 035431 STB CLINE UPDATE CLINE
1147*
1148 22761 021661 LDA T9
1149 22762 070070 SIA *+1
1150 22763 070517 LDA A,I
1151 22764 050065 AND OPMSK (076000)
1152 22765 013777 CPA OPCOD TEXT?
1153 22766 066771 JMP *+3 YES
1154 22767 025472 LDB TEMPS
1155 22770 166014 JMP LISTS,I
1156*
1157 22771 021660 LDA T8 GET NO-LINE-NUMBERS FLAG
1158 22772 072110 RZA *+2
1159 22773 160210 JSM OUTIA,I OUTPUT LINE #
1160 22774 121661 LDA T9,I
1161 22775 045661 ISZ T9
1162 22776 025661 LDB T9
1163 22777 050104 AND B177 CALCULATE CHARACTER COUNT
1164 23000 000134 ADA M3
1165 23001 070037 ADB A
1166 23001 070744 SAL 1
1167 23003 074537 LDB B,I
1168 23004 074404 SBL 8
1169 23005 074110 SZN *+2
1170 23006 070070 SIA *+1
1171 23007 125661 LDB T9 CALCULATE CHARACTER POINTER
1172 23010 074744 SBL 1
1173 23011 074070 SIB *+1
1174 23012 065335 JMP NPWRT OUTPUT TO BUFFER

```


PAGE 419 TERMINAL OPTION BLOCK

```

1176*
1177* LIST ROUTINE
1178*
1179 23013 061055 XLIST JSM GNEXT GET NEXT CHARACTER
1180 23014 013062 CPA B130 "X"?
1181 23015 067051 JMP XLIS3 YES
1182 23016 013063 CPA B43 "#"?
1183 23017 067053 JMP XLIS4 YES
1184 23020 055400 DSZ BADDR NO, RESET POINTER
1185 23021 162013 XLIS1 JSM GETLN,I GET OPTIONAL LINE NUMBERS
1186 23022 020012 LDA .1
1187 23023 024056 LDB .1E4
1188 23024 000132 ADA M1
1189 23025 031431 STA CLINE
1190 23026 035656 STB T6 SAVE ENDING LINE NUMBER
1191 23027 025424 LDB PBUFF CHECK FOR NULL PROGRAM
1192 23030 015423 CPB PBPTR
1193 23031 066130 JMP RE0
1194*
1195 23032 062731 XLIS2 JSM DNRWS OUTPUT LINE TO DISPLAY BUFFER
1196 23033 025431 LDB CLINE
1197 23034 074076 TCB
1198 23035 005656 ADB T6 T6—CLINE
1199 23036 074152 SBM *+3
1200 23037 025442 LDB STPFL CHECK STOP FLAG
1201 23040 014040 CPB .31
1202 23041 067064 JMP LOUT
1203 23042 021651 LDA T1
1204 23043 011447 CPA SLCOD CHECK SELECT CODES
1205 23044 067057 JMP XLIS5
1206*
1207 23045 060373 JSM ICNT GET CHARACTER COUNT
1208 23046 061337 JSM NDWRT PRINT THE LINE
1209 23047 061344 JSM CRLF1
1210 23050 067032 JMP XLIS2
1211*
1212 23051 045660 XLIS3 ISZ T8 SET NO-LINE-NUMBERS FLAG
1213 23052 067013 JMP XLIST
1214 23053 060664 XLIS4 JSM LNUM FETCH SELECT CODE
1215 23054 074204 SBL 12
1216 23055 035447 STB SLCOD
1217 23056 067021 JMP XLIS1
1218*
1219 23057 062542 XLIS5 JSM MTIB MOVE RECORD TO I/O BUFFER
1220 23060 063246 JSM MDRV CALL MODEM DRIVER
1221 23061 067032 JMP XLIS2
1222*
1223 23062 000130 B130 OCT 130
1224 23063 000043 B43 OCT 43
1225*
1226 23064 024127 LOUT LDB B170K
1227 23065 035447 STB SLCOD
1228 23066 066130 JMP RE0

```

PAGE 420

TERMINAL OPTION BLOCK

```

1230*
1231*   COMPILE ROUTINE
1232*
1233   23067  020132  XCOMP  LDA  MI
1234   23070  031433          STA  TFLAG
1235   23071  020041          LDA  .32
1236   23072  031707          STA  BLANK
1237   23073  162013          JSM  GETLN,I  GET OPTIONAL LINE NUMBERS
1238   23074  020012          LDA  .1
1239   23075  024056          LDB  .1E4
1240   23076  000132          ADA  MI
1241   23077  031431          STA  CLINE
1242   23100  035656          STB  T6      SAVE ENDING LINE NUMBER
1243   23101  025424          LDB  PBUFF  CHECK FOR NULL PROGRAM
1244   23102  015423          CPB  PBPTR
1245   23103  067117          JMP  XCOM2  DONE
1246*
1247   23104  062731  XCOM1  JSM  DNRWS  OUTPUT LINE TO DISPLAY BUFFER
1248   23105  025431          LDB  CLINE
1249   23106  074076          TCB
1250   23107  005656          ADB  T6      T6—CLINE
1251   23110  074352          SBM  XCOM2
1252*
1253   23111  062542          JSM  MTIB  MOVE LINE TO I/O BUFFER
1254   23112  023646          LDA  AT1
1255   23113  027647          LDB  AT1S
1256   23114  170004          XFR
1257   23115  170004          XFR
1258   23116  064357          JMP  RETN2  GO COMPILE
1259*
1260   23117  070742  XCOM2  SAR  16
1261   23120  031433          STA  TFLAG
1262   23121  066130          JMP  RE0

```

PAGE 421. TERMINAL OPTION BLOCK

```

1264*
1265* PROCESS DEFINED KEY
1266*
1267 23122 021661 KEYPR LDA T9 CHECK FOR SPECIAL FUNCTION
1268 23123 010022 CPA .5
1269 23124 067212 JMP SETAS SET ASCII—SHIFT
1270 23125 010023 CPA .6
1271 23126 067215 JMP SETAC SET ASCII—CNTRL
1272 23127 010024 CPA .7
1273 23130 067226 JMP PRST SET PRINT/STORE
1274 23131 010025 CPA .8
1275 23132 066642 JMP EVOD CHANGE PARITY
1276 23133 010026 CPA .9
1277 23134 067237 JMP XMIT TRANSMIT
1278*
1279 23135 024166 KEY1 LDB FWAM SEARCH FOR KEY, STARTING AT FWAM
1280 23136 015405 CPB FWUP
1281 23137 066043 JMP WHAT UNDEFINED
1282 23140 074517 LDA B,I
1283 23141 074070 SIB *+1
1284 23142 011661 CPA T9
1285 23143 067146 JMP *+3 FOUND IT
1286 23144 074437 ADB B,I
1287 23145 067136 JMP KEY1 KEEP SEARCHING
1288*
1289 23146 074517 LDA B,I
1290 23147 074070 SIB *+1
1291 23150 010012 CPA .1 NULL DEFINITION?
1292 23151 066043 JMP WHAT YES
1293 23152 031466 STA LOCL1 NO, SAVE LENGTH WORD
1294 23153 074517 LDA B,I
1295 23154 074070 SIB *+1
1296 23155 070110 SZA *+2 IS THIS A TYPING AID?
1297 23156 066043 JMP WHAT NO
1298 23157 074744 SBL 1 YES
1299 23160 035740 STB TMP SAVE CHARACTER POINTER
1300 23161 021466 LDA LOCL1
1301 23162 070744 SAL 1
1302 23163 000136 ADA M5 ASSUME ODD COUNT
1303 23164 031473 STA CCNT SAVE CHARACTER COUNT
1304 23165 070037 ADB A
1305 23166 074517 LDA B,I LOOK AT LAST WORD
1306 23167 050105 AND B377
1307 23170 070450 SZA *+9
1308 23171 045473 ISZ CONT CORRECT COUNT
1309 23172 010100 CPA ASTR1 CHECK FOR *
1310 23173 067175 JMP *+2
1311 23174 066201 JMP R46
1312 23175 074517 LDA B,I
1313 23176 050157 AND M256
1314 23177 040027 IOR AIEX1 IEX CODE
1315 23200 067206 JMP *+6
1316 23201 074517 LDA B,I
1317 23202 013210 CPA ASTR2 CHECK FOR *
1318 23203 067205 JMP *+2
1319 23204 066201 JMP R46

```

PAGE 422

TERMINAL OPTION BLOCK

| | | | | | | |
|-------|-------|--------|-------|-----|-------------|---------------------------|
| 1320 | 23205 | 023211 | | LDA | AIEX2 | IEX CODE |
| 1321 | 23206 | 074557 | | STA | B,I | |
| 1322 | 23207 | 066201 | | JMP | R46 | EXECUTE IMMEDIATELY |
| 1323* | | | | | | |
| 1324 | 00100 | | ASTR1 | EQU | B52 | |
| 1325 | 23210 | 025000 | ASTR2 | OCT | 025000 | |
| 1326 | 00027 | | AIEX1 | EQU | .11 | |
| 1327 | 23211 | 005400 | AIEX2 | OCT | 005400 | |
| 1328* | | | | | | |
| 1329* | | | | | | |
| 1330 | 23212 | 024033 | SFTAS | LDB | B20 | |
| 1331 | 23213 | 035662 | | STB | SHFLG | |
| 1332 | 23214 | 067217 | | JMP | *+3 | |
| 1333 | 23215 | 024054 | SETAC | LDB | .63 | |
| 1334 | 23216 | 035663 | | STB | CTFLG | |
| 1335 | 23217 | 162012 | | JSM | KEYIN,I | |
| 1336 | 23220 | 010105 | | CPA | B377 | DUMP I/O BUFFER? |
| 1337 | 23221 | 066346 | | JMP | DIOB | YES |
| 1338 | 23222 | 066210 | | JMP | R26 | |
| 1339* | | | | | | |
| 1340 | 23223 | 002450 | MSSG1 | DEF | *+*+2-ID-ID | |
| 1341 | 23224 | 052105 | | ASC | 2,TEXT | |
| | 23225 | 054124 | | | | |
| 1342* | | | | | | |
| 1343 | 23226 | 160170 | PRST | JSM | CLERA,I | CLEAR THE DISPLAY |
| 1344 | 23227 | 020062 | | LDA | FLGBT | |
| 1345 | 23230 | 031655 | | STA | T5 | |
| 1346 | 23231 | 027234 | | LDB | STMS | |
| 1347 | 23232 | 020021 | | LDA | 4 | |
| 1348 | 23233 | 066145 | | JMP | R10 | |
| 1349* | | | | | | |
| 1350 | 23234 | 002472 | STMS | DEF | *+*+2-ID-ID | |
| 1351 | 23235 | 051501 | | ASC | 2,SAVE | |
| | 23236 | 053105 | | | | |
| 1352* | | | | | | |
| 1353 | | | | | | |
| 1354 | 23237 | 062542 | XMIT | JSM | MTIR | MOVE RECORD TO I/O BUFFER |
| 1355 | 23240 | 060373 | | JSM | ICNT | GET CHARACTER COUNT |
| 1356 | 23241 | 010012 | | CPA | .1 | |
| 1357 | 23242 | 066134 | | JMP | REI | IGNORE NULL RECORD |
| 1358* | | | | | | |
| 1359 | 23243 | 062552 | | JSM | CRIT | |
| 1360 | 23244 | 063246 | | JSM | MDVR | CALL MODEM |
| 1361 | 23245 | 066134 | | JMP | REI | |

531

524

1,444,141

524

PAGE 423

TERMINAL OPTION BLOCK

```

1363*
1364*  MODEM OUTPUT DRIVER
1365*
1366  23246  021651  MDRV  LDA  TI      GET SELECT CODE
1367  23247  043641  IOR  B4001
1368  23250  063625  JSM  IOST    WAIT FOR MODEM TO BE READY
1369  23251  170402  RET                      STOP-FLAG RETURN
1370*
1371  23252  172741  STF  1      TURN OFF INTERRUPT
1372  23253  020211  LDA  BUFA    SET TO START OF I/O BUFFER
1373  23254  031400  STA  BADDR
1374*
1375  23255  021400  MDR1  LDA  BADDR
1376  23256  025400  LDB  BADDR
1377  23257  070002  SAR  1
1378  23260  070517  LDA  A,I    FETCH TWO CHARACTERS
1379  23261  074111  SLR  *+2
1380  23262  070346  RAR  8
1381  23263  070342  SAR  8
1382  23264  010031  CFA  EOL    IS IT FOL?
1383  23265  067271  JMP  MDR3   YES
1384  23266  063277  JSM  MDRS   NO
1385  23267  045400  ISZ  BADDR
1386  23270  067255  JMP  MDR1
1387*
1388  23271  020031  MDR3  LDA  CRTN  OUTPUT CRTN
1389  23272  063277  JSM  MDRS
1390  23273  020174  LDA  B177   OUTPUT RUBOUT
1391  23274  063277  JSM  MDRS
1392  23275  173741  CLF  1
1393  23276  170402  RET

```

PAGE 424

TERMINAL OPTION BLOCK

1395*

1396*

SERIALIZING SUBROUTINE

1397*

| | | | | | |
|------|-------|--------|------|-----|-------|
| 1398 | 23277 | 031661 | MDRS | STA | T9 |
| 1399 | 23300 | 025653 | | LDB | T3 |
| 1400 | 23301 | 070111 | | SLA | *+2 |
| 1401 | 23302 | 004062 | | ADB | FLGBT |
| 1402 | 23303 | 070002 | | SAR | 1 |
| 1403 | 23304 | 073650 | | RZA | *-3 |
| 1404 | 23305 | 021661 | | LDA | T9 |
| 1405 | 23306 | 074342 | | SBR | 8 |
| 1406 | 23307 | 074017 | | ADA | B |
| 1407 | 23310 | 040110 | | IOR | B1400 |
| 1408 | 23311 | 070744 | | SAL | 1 |
| 1409 | 23312 | 031661 | | STA | T9 |
| 1410 | 23313 | 025652 | | LDB | T2 |
| 1411 | 23314 | 004106 | | ADB | B777 |

INCLUDE PARITY

INCLUDE STOP BITS

 SET UP BIT COUNTER
 SWITCH TO SINGLE STOP BIT @ 130
 BAUD

| | | | | | |
|------|-------|--------|------|-----|-------|
| 1412 | 23315 | 074700 | | ABR | 15 |
| 1413 | 23316 | 004141 | | ADB | M10 |
| 1414 | 23317 | 035470 | | STB | L2T1 |
| 1415 | 23320 | 041651 | MDR2 | IOR | T1 |
| 1416 | 23321 | 040121 | | IOR | B4000 |
| 1417 | 23322 | 172141 | | OTA | 1 |
| 1418 | 23323 | 172601 | | STC | 1 |
| 1419 | 23324 | 172741 | | STF | 1 |
| 1420 | 23325 | 021661 | | LDA | T9 |
| 1421 | 23326 | 070002 | | SAR | 1 |
| 1422 | 23327 | 031661 | | STA | T9 |
| 1423 | 23330 | 025652 | | LDB | T2 |
| 1424 | 23331 | 076030 | | RIB | * |
| 1425 | 23332 | 045470 | | ISZ | L2T1 |
| 1426 | 23333 | 067320 | | JMP | MDR2 |
| 1427 | 23334 | 170402 | | RET | |

"OR" IN SELECT CODE

"OR" IN SO3

LOAD I/O REGISTER

OUTPUT THE DATA

SHIFT DATA RIGHT

SAVE SHIFTED DATA

LOAD TIMING CONSTANT

LOOP FOR TIMING

INCREMENT BIT COUNTER

PAGE 425

TERMINAL OPTION BLOCK

```

1429*
1430 SERVICE MODEM INTERRUPTS
1431*
1432 23335 040377 INTAD DEF 40377B ADDRESS OF INTERRUPT TABLE—1
1433 23336 00137 INTSA DEF *+1—ID
1434*
1435 23337 074742 SBR 16 INITIALIZE DATA WORD
1436 23340 023643 LDA M8 SET UP BIT COUNTER
1437 23341 031446 STA ADRSS
1438 23342 021652 LDA T2
1439 23343 070000 AAR 1 LOAD HALF-WAIT
1440 23344 000042 ADA .33 ALLOW FOR DELAY IN INTERRUPT ROUTINE
1441 23345 072030 RIA * LOOP FOR TIMING
1442*
1443 23346 021651 MDUI LDA T1 GET SELECT CODE
1444 23347 043641 IOR B4001
1445 23350 172141 OTA 1
1446 23351 172601 STC 1 SAMPLE
1447 23352 172741 STF 1
1448 23353 172241 LIA 1
1449 23354 074002 SBR 1 POSITION OLD DATA WORD
1450 23355 070111 SLA *+2
1451 23356 004062 ADB FLGBT
1452 23357 021652 LDA T2 LOAD TIMING CONSTANT
1453 23360 072030 RIA * LOOP FOR TIMING
1454 23361 045446 ISZ ADRSS
1455 23362 067346 JMP MDUI
1456 23363 074402 SBR 9
1457 23364 035446 STB ADRSS SAVE CODE
1458*
1459 23365 021655 LDA T5
1460 23366 070110 SZA *+2
1461 23367 067507 JMP MDSS
1462*
1463 23370 014031 CPB CRTN
1464 23371 067463 JMP SCRN
1465 23372 004151 ADB M32 CODE < 40R?
1466 23373 074152 SBM *+3 YES
1467 23374 004154 ADB M64 CODE < 140B?
1468 23375 074112 SBM *+2 YES
1469 23376 067450 JMP INT2
1470*
1471 23377 021657 LDA T7 OK TO CLEAR DISPLAY?
1472 23400 072210 RZA *+4
1473 23401 160170 JSM CLERA,I CLEAR THE DISPLAY
1474 23402 020211 LDA .BUFA
1475 23403 031657 STA T7
1476*
1477 23404 021400 LDA BADDR STASH IN DISPLAY BUFFER
1478 23405 010236 CPA SLEND
1479 23406 067450 JMP INT2 AT END OF BUFFER
1480 23407 025400 LDB BADDR
1481 23410 045400 ISZ BADDR
1482 23411 074006 RBR 1
1483 23412 070002 SAR 1
1484 23413 070517 LDA A,I FETCH WORD IN BUFFER

```

PAGE 426

TERMINAL OPTION BLOCK

| | | | | | | |
|-------|-------|--------|------|-----|-------|------------------------|
| 1485 | 23414 | 074112 | | SBM | *+2 | |
| 1486 | 23415 | 070346 | | RAR | 8 | |
| 1487 | 23416 | 050157 | | AND | M256 | |
| 1488 | 23417 | 041446 | | IOR | ADRSS | |
| 1489 | 23420 | 074132 | | SBM | *+2,C | |
| 1490 | 23421 | 070346 | | RAR | 8 | |
| 1491 | 23422 | 074577 | | STA | B,I | |
| 1492* | | | | | | |
| 1493 | 23423 | 021400 | | LDA | BADDR | SCROLL IF NECESSARY |
| 1494 | 23424 | 000151 | | ADA | M32 | |
| 1495 | 23425 | 070076 | | TCA | | |
| 1496 | 23426 | 001402 | | ADA | OTPTR | |
| 1497 | 23427 | 070113 | | SAP | *+2 | |
| 1498 | 23430 | 045402 | | ISZ | OTPTR | |
| 1499* | | | | | | |
| 1500 | 23431 | 021657 | MDU2 | LDA | T7 | STASH IN I/Q BUFFER |
| 1501 | 23432 | 013506 | | CPA | IOEND | |
| 1502 | 23433 | 067450 | | JMP | INT2 | AT END OF BUFFER |
| 1503 | 23434 | 025657 | | LDB | T7 | |
| 1504 | 23435 | 045657 | | ISZ | T7 | |
| 1505 | 23436 | 074006 | | RBR | 1 | |
| 1506 | 23437 | 070002 | | SAR | 1 | |
| 1507 | 23440 | 070517 | | LDA | A,I | FETCH WORD IN BUFFER |
| 1508 | 23441 | 074112 | | SBM | *+2 | |
| 1509 | 23442 | 070346 | | RAR | 8 | |
| 1510 | 23443 | 050157 | | AND | M256 | |
| 1511 | 23444 | 041446 | | IOR | ADRSS | |
| 1512 | 23455 | 074132 | | SBM | *+2,C | |
| 1513 | 23446 | 070346 | | RAR | 8 | |
| 1514 | 23447 | 074557 | | STA | B,I | |
| 1515* | | | | | | |
| 1516 | 23450 | 021651 | INT2 | LDA | T1 | GET SELECT CODE |
| 1517 | 23451 | 043641 | | IOR | B4001 | |
| 1518 | 2345 | 172141 | | OTA | 1 | |
| 1519 | 23453 | 172601 | | STC | 1 | |
| 1520 | 23454 | 172741 | | STF | 1 | |
| 1521 | 23455 | 172241 | | LIA | 1 | |
| 1522 | 23456 | 071511 | | SLA | INT2 | WAIT FOR STOP BIT |
| 1523 | 23457 | 021444 | | LDA | SAVEA | RESTORE A-REG |
| 1524 | 23460 | 025445 | | LDB | SAVEB | RESTORE B-REG |
| 1525 | 23461 | 173741 | | CLF | 1 | RESTORE INTERRUPT |
| 1526 | 23462 | 170402 | | RET | | |
| 1527* | | | | | | |
| 1528 | 23463 | 020105 | SCRN | LDA | B377 | |
| 1529 | 23464 | 031443 | | STA | KEYFG | SAY TO DUMP I/O BUFFER |
| 1530* | | | | | | |
| 1531 | 23465 | 021657 | | LDA | T7 | PUT EOL IN I/O BUFFER |
| 1532 | 23466 | 072110 | | RZA | *+2 | |
| 1533 | 23467 | 020211 | | LDA | .BUFA | |
| 1534 | 23470 | 070137 | | LDB | A | |
| 1535 | 23471 | 074006 | | RBR | 1 | |
| 1536 | 23472 | 070002 | | SAR | 1 | |
| 1537 | 23473 | 070517 | | LDA | A,I | |
| 1538 | 23474 | 074112 | | SBM | *+2 | |
| 1539 | 23475 | 070346 | | RAR | 8 | |
| 1540 | 23476 | 050157 | | AND | M256 | |

546

PAGE 427 TERMINAL OPTION BLOCK

| | | | | | | |
|-------|----------------------|--------|-------|-----|----------------|----------------------|
| 1541 | 23477 | 040031 | | ICR | EOL | |
| 1542 | 23500 | 074132 | | SBM | *+2,C | |
| 1543 | 23501 | 070346 | | RAR | 8 | |
| 1544 | 23502 | 074557 | | STA | B,I | |
| 1545 | 23503 | 070742 | | SAR | 16 | |
| 1546 | 23504 | 031657 | | STA | T7 | |
| 1547 | 23505 | 067450 | | JMP | INT2 | |
| 1548* | | | | | | |
| 1549 | 23506 | 003320 | IOEND | DEF | IOBUF+IOBUF+80 | |
| 1550* | | | | | | |
| 1551* | | | | | | |
| 1552* | STORE FROM INTERRUPT | | | | | |
| 1553* | | | | | | |
| 1554 | 23507 | 014031 | MDSS | CPB | CRTN | |
| 1555 | 23510 | 067606 | | JMP | MDS5 | |
| 1556 | 23511 | 004151 | | ADB | M32 | CODE < 40B? |
| 1557 | 23512 | 074152 | | SBM | *+3 | YES |
| 1558 | 23513 | 004154 | | ADB | M64 | CODE < 140B? |
| 1559 | 23514 | 074112 | | SBM | *+2 | YES |
| 1560 | 23515 | 067450 | | JMP | INT2 | |
| 1561* | | | | | | |
| 1562 | 23516 | 021657 | | LDA | T7 | |
| 1563 | 23517 | 072610 | | RZA | MDS2 | |
| 1564 | 23520 | 021446 | | LDA | ADRSS | PROCESS FIRST CODE |
| 1565 | 23521 | 060750 | | JSM | DIGCK | DIGIT? |
| 1566 | 23522 | 067530 | | JMP | MDS1 | NO |
| 1567 | 23523 | 035722 | | STB | INTGR | YES, SAVE IT |
| 1568 | 23524 | 024233 | | LDB | SBUFA | |
| 1569 | 23525 | 035706 | | STB | SBPTR | |
| 1570 | 23526 | 020133 | | LDA | M2 | |
| 1571 | 23527 | 067531 | | JMP | *+2 | |
| 1572 | 23530 | 020132 | MDS1 | LDA | M1 | |
| 1573 | 23531 | 031657 | | STA | T7 | |
| 1574 | 23532 | 067450 | | JMP | INT2 | |
| 1575* | | | | | | |
| 1576 | 23533 | 010132 | MDS2 | CPA | M1 | IGNORE? |
| 1577 | 23534 | 067450 | | JMP | INT2 | YES |
| 1578 | 23535 | 010133 | | CPA | M2 | IN LINE NUMBER? |
| 1579 | 23536 | 067557 | | JMP | MDS3 | |
| 1580 | 23537 | 021657 | MDST | LDA | T7 | NO, STORE CHARACTER |
| 1581 | 23540 | 010236 | | CPA | SLEND | |
| 1582 | 23541 | 067450 | | JMP | INT2 | AT END OF BUFFER |
| 1583 | 23542 | 025657 | | LDB | T7 | |
| 1584 | 23543 | 045657 | | ISZ | T7 | |
| 1585 | 23544 | 074006 | | RBR | 1 | |
| 1586 | 23545 | 070002 | | SAR | 1 | |
| 1587 | 23546 | 070517 | | LDA | A,I | FETCH WORD IN BUFFER |
| 1588 | 23547 | 074112 | | SBM | *+2 | |
| 1589 | 23550 | 070346 | | RAR | 8 | |
| 1590 | 23551 | 050157 | | AND | M256 | |
| 1591 | 23552 | 041446 | | IOR | ADRSS | |
| 1592 | 23553 | 074132 | | SBM | *+2,C | |
| 1593 | 23554 | 070346 | | RAR | 8 | |
| 1594 | 23555 | 074557 | | STA | B,I | |
| 1595 | 23556 | 067540 | | JMP | INT2 | |
| 1596* | | | | | | |

PAGE 428

TERMINAL OPTION BLOCK

| | | | | | | |
|-------|----------------------------|--------|------|------|---------|---------------------------|
| 1597 | 23557 | 021446 | MDS3 | LDA | ADRSS | PROCESS LINE NUMBER |
| 1598 | 23560 | 060750 | | JSM | DIGCK | DIGIT? |
| 1599 | 23561 | 067571 | | JMP | MDS4 | NO |
| 1600 | 23562 | 021722 | | LDA | INTGR | |
| 1601 | 23563 | 070704 | | SAL | 2 | |
| 1602 | 23564 | 001722 | | ADA | INTGR | |
| 1603 | 23565 | 070744 | | SAL | 1 | |
| 1604 | 23566 | 074017 | | ADA | B | |
| 1605 | 23567 | 013722 | | STA | INTGR | |
| 1606 | 23570 | 067450 | | JMP | INT2 | |
| 1607* | | | | | | |
| 1608 | 23571 | 021722 | MDS4 | LDA | INTGR | |
| 1609 | 23572 | 013427 | | STA | .LNUM | |
| 1610 | 23573 | 131706 | | STA | SBPTR,I | STORE LINE NUMBER |
| 1611 | 23574 | 045706 | | ISZ | SBPTR | |
| 1612 | 23575 | 023640 | | LDA | B5K | |
| 1613 | 23576 | 131706 | | STA | SBPTR,I | STORE SPECIAL OPCODE |
| 1614 | 23577 | 045706 | | ISZ | SBPTR | |
| 1615 | 23600 | 023777 | | LDA | OPCOD | |
| 1616 | 23601 | 131706 | | STA | SBPTR,I | STORE PAGE CODE |
| 1617 | 23602 | 024212 | | LDB | .SLBF | |
| 1618 | 23603 | 004022 | | ADB | .5 | |
| 1619 | 23604 | 035657 | | STB | T7 | |
| 1620 | 23605 | 067537 | | JMP | MDST | |
| 1621* | | | | | | |
| 1622 | 23606 | 021657 | MDS5 | LDA | T7 | PROCESS CRTN |
| 1623 | 23607 | 070550 | | SZA | MDS6 | |
| 1624 | 23610 | 010132 | | CPA | M1 | |
| 1625 | 23611 | 067622 | | JMP | MDS6 | |
| 1626 | 23612 | 010133 | | CPA | M2 | |
| 1627 | 23613 | 067622 | | JMP | MDS6 | |
| 1628 | 23614 | 021657 | | LDA | T7 | |
| 1629 | 23615 | 070002 | | .SAR | 1 | |
| 1630 | 23616 | 070070 | | SIA | *+1 | |
| 1631 | 23617 | 031706 | | STA | SBPTR | |
| 1632 | 23620 | 020105 | | LDA | B377 | |
| 1633 | 23621 | 031443 | | STA | KEYFG | |
| 1634 | 23622 | 070742 | MDS6 | SAR | 16 | |
| 1635 | 23623 | 031657 | | STA | T7 | |
| 1636 | 23624 | 067450 | | JMP | INT2 | |
| 1637* | | | | | | |
| 1638* | I/O STATUS ROUTINE | | | | | |
| 1639* | | | | | | |
| 1640* | LDA SELECT CODE (SAVED) | | | | | |
| 1641* | JSM IOST | | | | | |
| 1642* | | | | | | |
| 1643* | RETURN TO P+1 IF STOP FLAG | | | | | |
| 1644* | P+2 OTHERWISE | | | | | |
| 1645* | | | | | | |
| 1646 | 23625 | 025442 | IOST | LDR | STPFL | CHECK STOP FLAG |
| 1647 | 23626 | 014040 | | CPB | .31 | |
| 1648 | 23627 | 170402 | | RET | | |
| 1649* | | | | | | |
| 1650 | 23630 | 172741 | | STF | 1 | TURN OFF INTERRUPT SYSTEM |
| 1651 | 23631 | 172141 | | OTA | 1 | OUTPUT SELECT CODE |
| 1652 | 23632 | 172741 | | STF | 1 | LOAD DEVICE STATUS |

PAGE 429

TERMINAL OPTION BLOCK

| | | | | | |
|------|-------|--------|-----|-------|--------------------------|
| 1653 | 23633 | 177241 | LIB | I,C | AND TURN ON INTERRUPT |
| 1654 | 23634 | 074406 | RBR | 9 | |
| 1655 | 23635 | 075413 | SBP | IOST | LOOP IF BUSY (TESTS S10) |
| 1656 | 23636 | 064357 | JMP | RETN2 | |

PAGE 430

TERMINAL OPTION BLOCK

| | | | | | | |
|-------|--------------------------------------|--------|-------|-----|-------------|--------------------------|
| 1658 | 23637 | 001704 | BKPTA | DEF | BKPTR | |
| 1659 | 23640 | 005000 | B5K | OCT | 5000 | |
| 1660 | 23641 | 004001 | B4001 | OCT | 4001 | |
| 1661 | 23642 | 177752 | M22 | DEC | -22 | |
| 1662 | 23643 | 177770 | M8 | DEC | -8 | |
| 1663 | 23644 | 000112 | .74 | DEC | 74 | |
| 1664 | 23645 | 000141 | .97 | DEC | 97 | |
| 1665* | | | | | | |
| 1666 | 23646 | 001651 | ATI | DEF | T1 | |
| 1667 | 23647 | 040424 | ATIS | DEF | 40424B | |
| 1668 | 23650 | 001662 | TF1A | DEF | TF1 | |
| 1669 | 23651 | 001652 | TCON | DEF | *+1-ID | CONSTANT FOR TIMING LOOP |
| 1670 | 23652 | 002001 | | OCT | 002001 | 1014 |
| 1671 | 23653 | 062531 | | OCT | 062531 | 6559 |
| 1672 | 23654 | 021451 | | OCT | 021451 | 2329 |
| 1673 | 23655 | 104511 | | OCT | 104511 | 8949 |
| 1674* | | | | | | |
| 1675* | EDIT KEY TABLE (IMMEDIATE EXECUTION) | | | | | |
| 1676* | | | | | | |
| 1677 | 23656 | 000536 | EDITP | DEF | *+*+2-ID-ID | |
| 1678 | 23657 | 005600 | | OCT | 05600 | EXECUTE 13 0 |
| 1679 | 23660 | 006601 | | OCT | 06601 | STEP 15 1 |
| 1680 | 23661 | 010202 | | OCT | 10202 | RECALL 20 2 |
| 1681 | 23662 | 011203 | | OCT | 11203 | BACKSPACE 22 3 |
| 1682 | 23663 | 011604 | | OCT | 11604 | FORWARDSPACE 23 4 |
| 1683 | 23664 | 012205 | | OCT | 12205 | DOWNARROW 24 5 |
| 1684 | 23665 | 012606 | | OCT | 12606 | UPARROW 25 6 |
| 1685 | 23666 | 013207 | | OCT | 13207 | LEFTARROW 26 7 |
| 1686 | 23667 | 013610 | | OCT | 13610 | RIGHTARROW 27 8 |
| 1687 | 23670 | 015211 | | OCT | 15211 | CLEAR VAR 32 9 |
| 1688 | 23671 | 017612 | | OCT | 17612 | INSERT/DELETE 37 10 |
| 1689 | 23672 | 065213 | | OCT | 65213 | CLEAR 152 11 |
| 1690 | 23673 | 065614 | | OCT | 65614 | ACC 153 12 |
| 1691 | 23674 | 070215 | | OCT | 70215 | PRINT ALL 160 13 |
| 1692 | 23675 | 071316 | | OCT | 71316 | EOL 162 14 |

END

PAGE 431

TERMINAL OPTION BLOCK

1694*
1695* MULTI-CHARACTER KEY TABLE
1696*

| | | | MULTP | DEF | *+*+2-ID-ID | | |
|------|-------|--------|-------|-----|-------------|---------|--------|
| 1697 | 23676 | 003576 | | OCT | 05200 | LIST | 12 0 |
| 1698 | 23677 | 005200 | | OCT | 06201 | CONT | 14 1 |
| 1699 | 23700 | 006201 | | OCT | 07202 | TRACE | 16 2 |
| 1700 | 23701 | 007202 | | OCT | 07603 | RUN | 17 3 |
| 1701 | 23702 | 007603 | | OCT | 10604 | FETCH | 21 4 |
| 1702 | 23703 | 010604 | | OCT | 14205 | LOAD | 30 5 |
| 1703 | 23704 | 014205 | | OCT | 14606 | STORE | 31 6 |
| 1704 | 23705 | 014606 | | OCT | 17207 | NORMAL | 36 7 |
| 1705 | 23706 | 017207 | | OCT | 72210 | FIX | 164 8 |
| 1706 | 23707 | 072210 | | OCT | 72611 | FLT | 165 9 |
| 1707 | 23710 | 072611 | | OCT | 73212 | SCRATCH | 166 10 |
| 1708 | 23711 | 073212 | | OCT | 73613 | AUTO | 167 11 |
| 1709 | 23712 | 073613 | | OCT | 77714 | EXIT | 177 12 |
| 1710 | 23713 | 077714 | | | | | END |

1711*
1712* MULTI-CHARACTER MNEMONIC TABLE
1713*

| | | | MULTN | DEF | *+*+2-ID-ID | | |
|------|-------|--------|-------|-----|-------------|--------|--|
| 1714 | 23714 | 003632 | | OCT | 046111 | L I | |
| 1715 | 23715 | 046111 | | OCT | 051524 | S T | |
| 1716 | 23716 | 051524 | | OCT | 100103 | (0) C | |
| 1717 | 23717 | 100103 | | OCT | 047516 | O N | |
| 1718 | 23720 | 047516 | | OCT | 052201 | T (1) | |
| 1719 | 23721 | 052201 | | OCT | 052122 | T R | |
| 1720 | 23722 | 052122 | | OCT | 040503 | A C | |
| 1721 | 23723 | 040503 | | OCT | 042602 | E (2) | |
| 1722 | 23724 | 042602 | | OCT | 051125 | R U | |
| 1723 | 23725 | 051125 | | OCT | 047203 | N (3) | |
| 1724 | 23726 | 047203 | | OCT | 043105 | F E | |
| 1725 | 23727 | 043105 | | OCT | 052103 | T C | |
| 1726 | 23730 | 052103 | | OCT | 044204 | H (4) | |
| 1727 | 23731 | 044204 | | OCT | 046117 | L O | |
| 1728 | 23732 | 046117 | | OCT | 040504 | A D | |
| 1729 | 23733 | 040504 | | OCT | 102523 | (5) S | |
| 1730 | 23734 | 102523 | | OCT | 052117 | T O | |
| 1731 | 23735 | 052117 | | OCT | 051105 | R E | |
| 1732 | 23736 | 051105 | | OCT | 103116 | (6) N | |
| 1733 | 23737 | 103116 | | OCT | 047522 | O R | |
| 1734 | 23740 | 047522 | | OCT | 046501 | M A | |
| 1735 | 23741 | 046501 | | OCT | 046207 | L (7) | |
| 1736 | 23742 | 046207 | | OCT | 043111 | F I | |
| 1737 | 23743 | 043111 | | OCT | 054210 | X (8) | |
| 1738 | 23744 | 054210 | | OCT | 043114 | F L | |
| 1739 | 23745 | 043114 | | OCT | 052211 | T (9) | |
| 1740 | 23746 | 052211 | | OCT | 051503 | S C | |
| 1741 | 23747 | 051503 | | OCT | 051101 | R A | |
| 1742 | 23750 | 051101 | | OCT | 052103 | T C | |
| 1743 | 23751 | 052103 | | OCT | 044212 | H (10) | |
| 1744 | 23752 | 044212 | | OCT | 040525 | A U | |
| 1745 | 23753 | 040525 | | OCT | 052117 | T O | |
| 1746 | 23754 | 052117 | | OCT | 105505 | (11) E | |
| 1747 | 23755 | 105505 | | OCT | 054111 | X I | |
| 1748 | 23756 | 054111 | | OCT | 052214 | T (12) | |
| 1749 | 23757 | 052214 | | | | | |



532

1,444,141

532

PAGE 432 TERMINAL OPTION BLOCK

| | | | | | | |
|------|-------|--------|-----|--------|------|------|
| 1750 | 23760 | 041517 | OCT | 041517 | C O | |
| 1751 | 23761 | 046520 | OCT | 046520 | M P | |
| 1752 | 23762 | 146400 | OCT | 146400 | (13) | END. |

PAGE 433 TERMINAL OPTION BLOCK

| | | | | | | |
|-------|--------|--------|-------|-----|-------------|---|
| 1754 | 23763 | 160205 | ERRX | JSM | AERRA,I | MODE ERROR |
| 1755 | 23764 | 177661 | | DEC | -79 | |
| 1756* | | | | | | |
| 1757 | 23765 | 003754 | WHAM | DEF | *+*+2-ID-ID | |
| 1758 | 23766 | 053510 | | ASC | 3,WHAT? | |
| | 23767 | 040524 | | | | |
| | 23770 | 037440 | | | | |
| 1759 | 000000 | | LEFF | EQU | 23771B-* | |
| 1760* | | | | | | |
| 1761 | 01651 | | | ORG | 1651B | |
| 1762* | | | | | | |
| 1763 | 01740 | | TMP | EQU | XC | |
| 1764 | 01473 | | CCNT | EQU | TEMPS+1 | |
| 1765 | 01704 | | BKPTR | EQU | WIDTH | |
| 1766* | | | | | | |
| 1767 | 01651 | 000000 | T1 | NOP | | SELECT CODE |
| 1768 | 01652 | 000000 | T2 | NOP | | TIMING CONSTANT |
| 1769 | 01653 | 000000 | T3 | NOP | | PARITY INDICATOR |
| 1770 | 01654 | 000000 | T4 | NOP | | PAGE INFO |
| 1771 | 01655 | 000000 | T5 | NOP | | PRINT/STORE FLAG |
| 1772 | 01656 | 000000 | T6 | NOP | | FOR LISTING |
| 1773 | 01657 | 000000 | T7 | NOP | | OK/NOT OK TO CLEAR DISPLAY ON INTERRUPT |
| 1774 | 01660 | 000000 | T8 | NOP | | |
| 1775 | 01661 | 000000 | T9 | NOP | | NO-LINE-NUMBERS FLAG |
| 1776* | | | | | | |
| 1777 | 01662 | 000000 | TF1 | BSS | 4 | FLOATING TEMPORARY |
| 1778 | 01662 | | SHFLG | EQU | TF1 | |
| 1779 | 01663 | | CTFLG | EQU | TF1+1 | |
| 1780* | | | | | | |
| 1781 | 23771 | | | ORG | 23771B | |
| 1782* | | | | | | |
| 1783 | 23771 | 177772 | | ABS | ERRX-* | EXECUTION JUMP TABLE |
| 1784 | 23772 | 000000 | | NOP | | ** CHECKSUM WORD ** |
| 1785* | | | | | | |
| 1786 | 23773 | 176014 | | ABS | BN777-* | NON-FORMULA OPERATOR TABLE (NULL TABLE) |
| 1787 | 23774 | 176013 | | ABS | BN777-* | FUNCTION-NAME TABLE (NULL TABLE) |
| 1788 | 23775 | 176004 | | ABS | COMTB-* | COMMAND-NAME TABLE |
| 1789 | 23776 | 176010 | | ABS | SYSNT-* | STATEMENT-NAME TABLE |
| 1790 | 23777 | 016000 | OPCOD | OCT | 16000 | |
| 1791 | | | | END | | |

* NO ERRORS*

PAGE 434

CROSS-REFERENCE TABLE

| | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|
| .1 | 0020 | 0925 | 0945 | 0980 | 1087 | 1186 | 1238 | 1291 | 1356 |
| .10 | 0071 | 0072 | | | | | | | |
| .100 | 0070 | | | | | | | | |
| .1000 | 0069 | | | | | | | | |
| .11 | 0035 | 0043 | 1326 | | | | | | |
| .110 | 0569 | 0577 | | | | | | | |
| .12 | 0036 | | | | | | | | |
| .13 | 0039 | | | | | | | | |
| .15 | 0048 | 0083 | | | | | | | |
| .16 | 0049 | 0084 | | | | | | | |
| .1E4 | 0068 | 0576 | 1070 | 1123 | 1187 | 1239 | | | |
| .2 | 0027 | | | | | | | | |
| .21 | 0050 | | | | | | | | |
| .22 | 0051 | 0087 | | | | | | | |
| .23 | 0052 | 0088 | | | | | | | |
| .28 | 0053 | | | | | | | | |
| .3 | 0028 | 0045 | 1024 | 1059 | | | | | |
| .31 | 0054 | 0090 | 0723 | 1201 | 1647 | | | | |
| .32 | 0055 | 0736 | 0866 | 0915 | 1235 | | | | |
| .33 | 0056 | 1440 | | | | | | | |
| .34 | 0057 | 0091 | | | | | | | |
| .37 | 0058 | | | | | | | | |
| .4 | 0029 | 0573 | 0636 | 1042 | 1347 | | | | |
| .40 | 0059 | | | | | | | | |
| .41 | 0060 | 0958 | | | | | | | |
| .43 | 0061 | | | | | | | | |
| .45 | 0062 | | | | | | | | |
| .46 | 0063 | | | | | | | | |
| .48 | 0064 | | | | | | | | |
| .5 | 0030 | 0564 | 1268 | 1618 | | | | | |
| .58 | 0065 | 0094 | | | | | | | |
| .6 | 0031 | 0044 | 1270 | | | | | | |
| .63 | 0066 | 1333 | | | | | | | |
| .7 | 0032 | 1272 | | | | | | | |
| .74 | 1663 | 0713 | | | | | | | |
| .8 | 0033 | 0600 | 1274 | | | | | | |
| .9 | 0034 | 1276 | | | | | | | |
| .97 | 1664 | 0725 | | | | | | | |
| .99 | 0074 | | | | | | | | |
| .BUFA | 0180 | 0794 | 0829 | 0965 | 1372 | 1474 | 1533 | | |
| .FADA | 0185 | | | | | | | | |
| .FDVA | 0188 | 0597 | | | | | | | |
| .FMFA | 0187 | | | | | | | | |
| .FSBA | 0186 | | | | | | | | |
| .FSR | 0190 | | | | | | | | |
| .LNUM | 0314 | 0631 | 0661 | 0935 | 1609 | | | | |
| .NPWR | 0494 | | | | | | | | |
| .SLBF | 0181 | 0551 | 0667 | 0782 | 0848 | 0853 | 1617 | | |
| A | 0000 | 0612 | 0756 | 0803 | 1150 | 1165 | 1304 | 1378 | 1484 |
| | 1537 | 1587 | | | | | | 1507 | 1534 |
| ACTSA | 0174 | 0792 | 0947 | | | | | | |
| ADRSS | 0327 | 1437 | 1454 | 1457 | 1488 | 1511 | 1564 | 1591 | 1597 |
| AERRA | 0176 | 0581 | 1754 | | | | | | |
| AIEX1 | 1326 | 1314 | | | | | | | |
| AIEX2 | 1327 | 1320 | | | | | | | |

[illegible]

[illegible]

549

PAGE 437

CROSS-REFERENCE TABLE

| | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|
| ELINA | 0169 | | | | | | | |
| ENDS | 0480 | | | | | | | |
| ENTEA | 0218 | | | | | | | |
| ENTFA | 0189 | | | | | | | |
| EOL | 0038 | 0807 | 0832 | 0936 | 0944 | 0953 | 1382 | 1541 |
| EOL1 | 0928 | 0549 | | | | | | |
| EOLCK | 0483 | | | | | | | |
| EOLL | 0923 | 0707 | | | | | | |
| EOST | 0481 | | | | | | | |
| ERRA | 0177 | | | | | | | |
| ERRX | 1754 | 0515 | 1783 | | | | | |
| EVOD | 1034 | 1275 | | | | | | |
| EXP | 0401 | 0402 | | | | | | |
| EXPOP | 0118 | | | | | | | |
| F | 0081 | | | | | | | |
| FDAAA | 0222 | | | | | | | |
| FETCH | 0488 | | | | | | | |
| FILLA | 0201 | | | | | | | |
| FIRST | 0367 | | | | | | | |
| FIX | 0501 | | | | | | | |
| FLGBT | 0073 | 0540 | 1344 | 1401 | 1451 | | | |
| FLOAT | 0386 | | | | | | | |
| FLTFA | 0193 | | | | | | | |
| FLTRA | 0192 | 0598 | | | | | | |
| FLTSA | 0195 | | | | | | | |
| FN | 0148 | | | | | | | |
| FND | 0462 | 1131 | | | | | | |
| FND1A | 0224 | | | | | | | |
| FNDAA | 0223 | | | | | | | |
| FNDPS | 0476 | 1114 | 1124 | | | | | |
| FOPBS | 0206 | | | | | | | |
| FORD | 0862 | 0697 | | | | | | |
| FORMA | 0172 | | | | | | | |
| FORME | 0336 | | | | | | | |
| FORMS | 0335 | | | | | | | |
| FORMT | 0334 | | | | | | | |
| FPAGE | 0451 | | | | | | | |
| FROMA | 0122 | | | | | | | |
| FSCA | 0173 | | | | | | | |
| FSCE2 | 0441 | | | | | | | |
| FSCE4 | 0446 | | | | | | | |
| FTMP1 | 0337 | | | | | | | |
| FTMP2 | 0338 | | | | | | | |
| FW2 | 0865 | 0765 | | | | | | |
| FWAM | 0157 | 1279 | | | | | | |
| FWUP | 0296 | 1280 | | | | | | |
| FXFLA | 0194 | 0585 | | | | | | |
| GETAD | 0461 | | | | | | | |
| GETCH | 0471 | | | | | | | |
| GETLN | 0527 | 1086 | 1185 | 1237 | | | | |
| GETSK | 0475 | | | | | | | |
| GFLAG | 0404 | | | | | | | |
| GFRST | 0474 | 0929 | 0984 | | | | | |
| GNEXT | 0473 | 1179 | | | | | | |
| GPARM | 0424 | | | | | | | |
| GSIGN | 0486 | | | | | | | |

[illegible]

PAGE 439

CROSS-REFERENCE TABLE

[illegible]

CROSS-REFERENCE TABLE

[illegible]

CROSS-REFERENCE TABLE

[illegible]

554

CROSS-REFERENCE TABLE

| | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|
| RUNA | 0163 | 1071 | | | | | | | | | | |
| S | 0082 | | | | | | | | | | | |
| SALTA | 0207 | | | | | | | | | | | |
| SAVBP | 0452 | | | | | | | | | | | |
| SAVEA | 0325 | 1523 | | | | | | | | | | |
| SAVEB | 0326 | 1524 | | | | | | | | | | |
| SBADR | 0394 | | | | | | | | | | | |
| SBPTR | 0387 | 0939 | 0940 | 0942 | 1569 | 1610 | 1611 | 1613 | 1614 | 1616 | 1631 | |
| SBPUD | 0431 | | | | | | | | | | | |
| SBUFA | 0199 | 1568 | | | | | | | | | | |
| SCRN | 1528 | 1464 | | | | | | | | | | |
| SEC | 0320 | | | | | | | | | | | |
| SETAC | 1333 | 1271 | | | | | | | | | | |
| SETAS | 1330 | 1269 | | | | | | | | | | |
| SFLAG | 0393 | | | | | | | | | | | |
| SGNS | 0493 | | | | | | | | | | | |
| SHFLG | 1778 | 0676 | 0754 | 0758 | 1331 | | | | | | | |
| SIGN | 0399 | 0400 | | | | | | | | | | |
| SLCOD | 0328 | 0809 | 0817 | 1204 | 1216 | 1227 | | | | | | |
| SLEND | 0202 | 0641 | 0900 | 1478 | 1581 | | | | | | | |
| SLWST | 0447 | | | | | | | | | | | |
| SNFLA | 0197 | | | | | | | | | | | |
| SPLFA | 0196 | | | | | | | | | | | |
| SSBPT | 0395 | | | | | | | | | | | |
| SSYMA | 0170 | | | | | | | | | | | |
| ST | 0377 | | | | | | | | | | | |
| START | 0162 | 0007 | | | | | | | | | | |
| STEXP | 0443 | | | | | | | | | | | |
| STMS | 1350 | 1346 | | | | | | | | | | |
| STOR1 | 0428 | | | | | | | | | | | |
| STORE | 0427 | | | | | | | | | | | |
| STPFL | 0323 | 0644 | 1200 | 1646 | | | | | | | | |
| STPOP | 0430 | | | | | | | | | | | |
| STSRH | 0442 | | | | | | | | | | | |
| SUMPA | 0227 | | | | | | | | | | | |
| SYE12 | 0485 | | | | | | | | | | | |
| SYE25 | 0460 | | | | | | | | | | | |
| SYMC1 | 0433 | | | | | | | | | | | |
| SYMC2 | 0432 | | | | | | | | | | | |
| SYMCK | 0434 | | | | | | | | | | | |
| SYMTP | 0298 | | | | | | | | | | | |
| SYSNT | 0518 | 1789 | | | | | | | | | | |
| SYTSA | 0209 | | | | | | | | | | | |
| T1 | 1767 | 0575 | 0607 | 0618 | 1075 | 1203 | 1366 | 1415 | 1443 | 1516 | 1666 | |
| T2 | 1768 | 0601 | 1410 | 1423 | 1438 | 1452 | | | | | | |
| T3 | 1769 | 0615 | 1035 | 1038 | 1399 | | | | | | | |
| T4 | 1770 | 0592 | 0638 | 0686 | 0742 | 0749 | 0987 | 1061 | | | | |
| T5 | 1771 | 0594 | 0603 | 0610 | 0626 | 0790 | 1345 | 1459 | | | | |
| T6 | 1772 | 1133 | 1190 | 1198 | 1242 | 1250 | | | | | | |
| T7 | 1773 | 0648 | 1471 | 1475 | 1500 | 1503 | 1504 | 1531 | 1546 | 1562 | 1573 | |
| | 1580 | 1583 | 1584 | 1619 | 1622 | 1628 | 1635 | | | | | |
| T8 | 1774 | 0649 | 1157 | 1212 | | | | | | | | |
| T9 | 1775 | 0656 | 0664 | 0683 | 0714 | 0726 | 0729 | 0737 | 0740 | 0745 | 0747 | |
| | 0753 | 0757 | 1056 | 1135 | 1144 | 1145 | 1148 | 1160 | 1161 | 1162 | 1171 | |
| | 1267 | 1284 | 1398 | 1404 | 1409 | 1420 | 1422 | | | | | |
| TCON | 1669 | 0595 | | | | | | | | | | |

SSS

542

CROSS-REFERENCE TABLE

[illegible]

PAGE 444

CROSS-REFERENCE TABLE

XSCRA 0220
 XTRM 0572 0508
 YC 0414
 ZAP 0152
 ZONK 0490

PAGE 445

EXTENDED I/O BLOCK'S CONSTANTS, SYNTAX, AND LINKAGES

0508 LST
 0509* BASE PAGE EQUATES AND TEMPORARIES
 0510*
 0511 01641 ORG 1641B
 0512*
 0513 01641 000000 T5 BSS 1
 0514 01642 000000 T8 BSS 1
 0515 01643 000000 TBEND BSS 1
 0516 01644 000000 T1 BSS 1
 0517 01645 000000 BSS 1
 0518 01645 PTR1 EQU 1645B
 0519 01646 000000 RLFLG BSS 1
 0520 01647 000000 TW BSS 1
 0521 01647 TMPRE EQU 1647B
 0522 01650 000000 TX BSS 1
 0523 01651 000000 TD BSS 1
 0524 01652 000000 MF BSS 1
 0525 01653 000000 SD BSS 1
 0526 01654 000000 DP BSS 1
 0527 01654 TNULL EQU 1654B
 0528 01655 000000 EX BSS 1
 0529 01656 000000 T2 BSS 1
 0530 01657 000000 T3 BSS 1
 0531 01660 000000 T6 BSS 1
 0532 01661 000000 T7 BSS 1
 0533 01662 000000 T9 BSS 1
 0534 01663 000000 .FLP BSS 1
 0535*
 0536 01665 ORG 1665B
 0537 01665 000000 FLOOP BSS 1
 0538*
 0539* LOCAL TEMPORARIES FOR IMPLIED "FOR" LOOP IS AS FOLLOWS:
 0540*
 0541* WORD 1 ADDRESS OF VARIABLE
 0542* WORD 2 NAME OF VARIABLE
 0543* WORD 3 FINAL VALUE (INTEGER FORM)
 0544* WORD 4 STEP SIZE
 0545* WORD 5 ADDRESS OF 1ST WORD AFTER FOR STMT.
 0546*
 0547 01666 000000 .FLOOP BSS 5 FIRST LOOP BLOCK
 0548 01673 000000 BSS 5 SECOND LOOP BLOCK
 0549*
 0550 01700 000000 .BADR BSS 1
 0551 00051 ASC EQU 46
 0552 00047 ASCP EQU 43
 0553 00050 ASCM EQU 45
 0554 01712 .SIGN EQU 1712B
 0555 00074 .18 EQU B22

PAGE 446

EXTENDED I/O BLOCK'S CONSTANTS, SYNTAX, AND LINKAGES

| | | | | | |
|-------|---------------------|--------|-------|---------|---|
| 0557 | 22000 | | ORG | 22000B | |
| 0558* | | | | | |
| 0559* | FORMATTED I/O BLOCK | | | | |
| 0560* | | | | | |
| 0561 | 12001 | QUOTE | EQU | 12001B | |
| 0562 | 12007 | ASSOP | EQU | QUOTE+6 | |
| 0563* | | | | | |
| 0564* | | | | | |
| 0565 | 22000 | 000213 | ID | ABS | SOUT-* OUTPUT SYNTAX |
| 0566 | 22001 | 000117 | | ABS | SGET-* |
| 0567 | 22002 | 000270 | | ABS | CMDS-* |
| 0568 | 22003 | 000000 | | BSS | 1 |
| 0569* | | | | | |
| 0570 | 22004 | 041515 | SYSNT | ASC | 1,CMD |
| 0571 | 22005 | 042242 | | OCT | 42242 D; OPCODE 2 |
| 0572 | 22006 | 042516 | | ASC | 2,ENTER |
| | 22007 | 052105 | | | |
| 0573 | 22010 | 051243 | | OCT | 51243 R; OP CODE 3 |
| 0574 | 22011 | 047525 | | ASC | 3,OUTPUT |
| | 22012 | 052120 | | | |
| | 22013 | 052524 | | | |
| 0575 | 22014 | 162377 | NULL | OCT | 162377 OP CODE 4; NULL |
| 0576* | | | | | |
| 0577* | | | | | |
| 0578 | 22015 | 043117 | NFOPT | OCT | 043117 FO NON-FORMULA OPERATOR
TABLE |
| 0579 | 22016 | 051220 | | OCT | 051220 R (16) |
| 0580 | 22017 | 052117 | | OCT | 052117 TO |
| 0581 | 22020 | 110523 | | OCT | 110523 (17) S |
| 0582 | 22021 | 052105 | | OCT | 052105 TE |
| 0583 | 22022 | 050322 | | OCT | 50322 P (18) |
| 0584* | | | | | |
| 0585* | | | | | |
| 0586 | 22023 | 000331 | | ABS | EIAND-* |
| 0587 | 22024 | 000324 | | ABS | EIOR-* |
| 0588 | 22025 | 000313 | | ABS | EROT-* |
| 0589 | 22026 | 000441 | | ABS | STAT-* |
| 0590 | 22027 | 000452 | | ABS | CHAR-* |
| 0591 | 22030 | 000500 | | ABS | BIN-* |
| 0592 | 22031 | 000457 | | ABS | LIN-* |
| 0593 | 22032 | 000472 | | ABS | SPA-* |
| 0594 | 22033 | 000410 | | ABS | EOCT-* |
| 0595* | | | | | |
| 0596 | 22034 | 001661 | FLOP | DEF | .FLOP-5 |
| 0597 | 22035 | 150000 | .SC13 | OCT | 150000 |
| 0598 | 22036 | 177770 | M8 | DEC | -8 |
| 0599 | 22037 | 154000 | SC13 | OCT | 154000 |
| 0600 | 22040 | 001652 | AMF | DEF | MF |
| 0601 | 22041 | 000137 | AROW | OCT | 137 LEFT ARROW |
| 0602 | 22042 | 000176 | ALTM | OCT | 176 ALT MODE |
| 0603 | 22043 | 000033 | ESCM | OCT | 33 ESC MODE |
| 0604 | 22044 | 000105 | ASCE | OCT | 105 E |
| 0605 | 22045 | 031166 | B3116 | OCT | 31166 |
| 0606 | 22046 | 000120 | .80 | DEC | 80 |
| 0607 | 22047 | 002001 | B2001 | OCT | 2001 |
| 0608 | 22050 | 100700 | STEC | OCT | -77100 |
| 0609 | 22051 | 012025 | FMTOP | OCT | 12025 |

PAGE 447

EXTENDED I/O BLOCK'S CONSTANTS, SYNTAX, AND LINKAGES

| | | | | | | |
|-------|-------|--------|-------|-----|-------------------|----------|
| 0610 | 22052 | 000021 | .17 | DEC | 17 | |
| 0611 | 22053 | 007777 | BIT12 | OCT | 7777 | |
| 0612 | 22054 | 000032 | NFOPA | DEF | NFOPT+NFOPT-ID-ID | |
| 0613* | | | | | | |
| 0614* | | | | | | |
| 0615 | 22055 | 047503 | FUNNT | ASC | 1,OCT | |
| 0616 | 22056 | 052222 | | OCT | 52222 | T;22 |
| 0617 | 22057 | 051520 | | ASC | 1,SPA | |
| 0618 | 22060 | 040623 | | OCT | 40623 | A;23 |
| 0619 | 22061 | 046111 | | ASC | 1,LIN | |
| 0620 | 22062 | 047224 | | OCT | 47224 | N;24 |
| 0621 | 22063 | 041111 | | ASC | 1,BIN | |
| 0622 | 22064 | 047225 | | OCT | 47225 | N;25 |
| 0623 | 22065 | 041510 | | ASC | 2,CHAR | |
| | 22066 | 040522 | | | | |
| 0624 | 22067 | 113123 | | OCT | 113123 | 26;STAT |
| 0625 | 22070 | 052101 | | ASC | 1,TAT | |
| 0626 | 22071 | 052227 | | OCT | 52227 | T;27 |
| 0627 | 22072 | 051117 | | ASC | 1,ROT | |
| 0628 | 22073 | 052230 | | OCT | 52230 | T;30 |
| 0629 | 22074 | 044516 | | ASC | 2,INOR | |
| | 22075 | 047522 | | | | |
| 0630 | 22076 | 114502 | | OCT | 114502 | 31;BIAND |
| 0631 | 22077 | 044501 | | ASC | 2,IAND | |
| | 22100 | 047104 | | | | |
| 0632 | 22101 | 155000 | | OCT | 155000 | 32: NULL |

MORE ABSOLUTE LINKS TO THE BASIC SYSTEM

| | | | | | | |
|-------|-------|--------|-------|-----|--------|--------------------|
| 0633* | | | | | | |
| 0634* | | | | | | |
| 0635* | | | | | | |
| 0636* | | | | | | |
| 0637 | 22102 | 004304 | RITE | DEF | 4304B | |
| 0638 | 22103 | 007474 | MOUTA | DEF | 7474B | |
| 0639 | 22104 | 006656 | PRINN | DEF | 6656B | |
| 0640 | 22105 | 007366 | SFMT | DEF | 7366B | |
| 0641 | 22106 | 007303 | TYPEC | DEF | 7303B | |
| 0642 | 22107 | 007240 | QTCHK | DEF | 7240B | |
| 0643 | 22110 | 006773 | PFORM | DEF | 6773B | |
| 0644 | 22111 | 007007 | PRCHK | DEF | 7007B | |
| 0645 | 22112 | 007227 | CKEOL | DEF | 7227B | |
| 0646 | 22113 | 011444 | OPCHI | DEF | 11444B | |
| 0647 | 22114 | 007356 | FSPR | DEF | 7356B | |
| 0648 | 22115 | 012233 | VAR01 | DEF | 12233B | ADDRESS OF VAROP+2 |
| 0649 | 22116 | 016504 | IOHED | DEF | 16504B | |
| 0650 | 22117 | 007402 | PLOTE | DEF | 7402B | |



| | | | | | | |
|-------|-------|--------|----------------------|-----|---------|--------------------------------------|
| 0652* | | | | | | |
| 0653* | | | GET STATEMENT SYNTAX | | | |
| 0654* | | | | | | |
| 0655 | 22120 | 070742 | SGET | SAR | 16 | |
| 0656 | 22121 | 031656 | | STA | T2 | CLEAR NEST COUNTER |
| 0657 | 22122 | 021717 | | LDA | NUM | |
| 0658 | 22123 | 050065 | | AND | OPMSK | |
| 0659 | 22124 | 070744 | | SAL | I | |
| 0660 | 22125 | 031642 | | STA | T8 | SAVE PAGE INFO |
| 0661 | 22126 | 062247 | | JSM | STBLK | |
| 0662* | | | | | | |
| 0663 | 22127 | 062434 | | JSM | IOHD | GET (S.C., FMT#) |
| 0664* | | | | | | |
| 0665 | 22130 | 060513 | SGP1 | JSM | LPCK | LOOK FOR LEFT BRACKET |
| 0666 | 22131 | 066171 | | JMP | SGP4 | NOT FOUND |
| 0667 | 22132 | 061055 | | JSM | GNEXT | |
| 0668 | 22133 | 045706 | | ISZ | SBPTR | |
| 0669 | 22134 | 062422 | | JSM | NFOP | LOOK FOR "FOR" |
| 0670 | 22135 | 164307 | | JMP | E15+1,I | NOT FOUND |
| 0671 | 22136 | 010033 | | CPA | 16 | FOR? |
| 0672 | 22137 | 066141 | | JMP | *+2 | YES |
| 0673 | 22140 | 164307 | | JMP | E15+1,I | NO, ERROR |
| 0674 | 22141 | 045656 | | ISZ | T2 | INCREMENT NEST COUNTER |
| 0675 | 22142 | 160246 | | JSM | VAROA,I | GO LOOK FOR A VARIABLE |
| 0676 | 22143 | 164321 | | JMP | E25+1,I | NONE FOUND, ERROR |
| 0677 | 22144 | 164321 | | JMP | E25+1,I | SUBSCRIPT, ERROR |
| 0678 | 22145 | 060463 | | JSM | SYMC1 | FOUND ONE, LOOK FOR "=" |
| 0679 | 22146 | 012006 | | DEF | ASSOP-1 | |
| 0680 | 22147 | 164324 | | JMP | E25+4,I | |
| 0681 | 22150 | 160202 | | JSM | FSCA,I | GO LOOK FOR A FORMULA |
| 0682 | 22151 | 062422 | | JSM | NFOP | LOOK FOR "TO" |
| 0683 | 22152 | 164322 | | JMP | E25+2,I | NOT FOUND, ERROR |
| 0684 | 22153 | 012052 | | CPA | 17 | TO? |
| 0685 | 22154 | 066156 | | JMP | *+2 | YES |
| 0686 | 22155 | 164322 | | JMP | E25+2,I | NO, ERROR |
| 0687 | 22156 | 160202 | | JSM | FSCA,I | GO LOOK FOR A FORMULA |
| 0688 | 22157 | 062422 | | JSM | NFOP | LOOK FOR "STEP" |
| 0689 | 22160 | 066165 | | JMP | SGP2 | NOT FOUND |
| 0690 | 22161 | 010074 | | CPA | 18 | STEP? |
| 0691 | 22162 | 066164 | | JMP | *+2 | YES |
| 0692 | 22163 | 164323 | | JMP | E25+3,I | NO, ERROR |
| 0693 | 22164 | 160202 | | JSM | FSCA,I | GO LOOK FOR FORMULA |
| 0694 | 22165 | 060510 | SGP2 | JSM | COMCE | DEMAND A COMMA |
| 0695 | 22166 | 061055 | SGP3 | JSM | GNEXT | |
| 0696 | 22167 | 045706 | | ISZ | SBPTR | |
| 0697 | 22170 | 066130 | | JMP | SGP1 | |
| 0698* | | | | | | |
| 0699 | 22171 | 055400 | SGP4 | DSZ | BADDR | |
| 0700 | 22172 | 055706 | | DSZ | SBPTR | |
| 0701 | 22173 | 062233 | | JSM | SVARA | LOOK FOR A VARIABLE (STRING ALLOWED) |
| 0702 | 22174 | 010031 | BACK | CPA | EOL | END-OF-STATEMENT? |
| 0703 | 22175 | 066201 | | JMP | NESC | YES |
| 0704 | 22176 | 060504 | | JSM | COMCK | LOOK FOR A COMMA |
| 0705 | 22177 | 066205 | | JMP | SGP5 | NOT FOUND |
| 0706 | 22200 | 066166 | | JMP | SGP3 | |
| 0707* | | | | | | |

PAGE 449 EXTENDED I/O BLOCK'S CONSTANTS, SYNTAX, AND LINKAGES

| | | | | | |
|-------|---|--------|-------|-------------|--|
| 0708 | 22201 | 021656 | NESC | LDA T2 | CHECK NEST COUNTER |
| 0709 | 22202 | 072110 | | R2A *+2 | |
| 0710 | 22203 | 164203 | | JMP ACTSA,I | STATEMENT IS OK |
| 0711 | 22204 | 164301 | | JMP E10,I | |
| 0712* | | | | | |
| 0713 | 22205 | 060534 | SGP5 | JSM RPCKE | DEMAND A RIGHT BRACKET |
| 0714 | 22206 | 055656 | | DSZ T2 | DECREMENT NEST COUNTER |
| 0715 | 22207 | 066210 | | JMP *+1 | ALLOW FOR SKIP |
| 0716 | 22210 | 025656 | | LDB T2 | |
| 0717 | 22211 | 075153 | | SBP BACK | |
| 0718 | 22212 | 164302 | | JMP E10+1,I | |
| 0719* | | | | | |
| 0720* | OUTPUT STMT SYNTAX | | | | |
| 0721* | | | | | |
| 0722 | 22213 | 062247 | SOUT | JSM STBLK | |
| 0723 | 22214 | 072150 | | RZA *+3 | |
| 0724 | 22215 | 062434 | | JSM IOHD | STRING BLOCK NOT IN DO STRANDED SYNTAX |
| 0725 | 22216 | 166102 | | JMP RITE,I | |
| 0726 | 22217 | 045706 | | ISZ SBPTR | |
| 0727 | 22220 | 060520 | | JSM LPCKE-1 | STRING BLOCK IN, TRY INTERNAL CONVERSION |
| 0728 | 22221 | 161641 | | JSM T5,I | LOOK FOR STRING VARIABLE |
| 0729 | 22222 | 066230 | | JMP SOUT2 | P+1 |
| 0730 | 22223 | 066227 | | JMP SOUT1 | P+2 |
| 0731 | 22224 | 162116 | SOUT3 | JSM IOHED,I | P+3, STRING ARRAY NAME FOUND |
| 0732 | 22225 | 062435 | | JSM IOHD+1 | |
| 0733 | 22226 | 166102 | | JMP RITE,I | |
| 0734* | | | | | |
| 0735 | 22227 | 154110 | SOUT1 | DSZ BADRP,I | |
| 0736 | 22230 | 154110 | SOUT2 | DSZ BADRP,I | |
| 0737 | 22231 | 160202 | | JSM FSCA,I | CHECK SELECT CODE |
| 0738 | 22232 | 066224 | | JMP SOUT3 | |
| 0739* | | | | | |
| 0740* | CHECK SYNTAX ON NUMERIC OR STRING VARIABLES | | | | |
| 0741* | | | | | |
| 0742 | 22233 | 025641 | SVARA | LDB T5 | |
| 0743 | 22234 | 076250 | | RZB RECO2 | IS STRING BLOCK IN? |
| 0744 | 22235 | 160246 | | JSM VAROA,I | NO, ERROR IF ANY STRINGS |
| 0745 | 22236 | 164307 | | JMP E15+1,I | P+1, NO LETTER FOUND |
| 0746 | 22237 | 170402 | | RET | P+2, ARRAY VARIABLE FOUND AND STORED |
| 0747 | 22240 | 170402 | | RET | P+3, SIMPLE VARIABLE FOUND AND STORED |
| 0748* | | | | | |
| 0749 | 22241 | 161641 | RECO2 | JSM T5,I | YES, TRY STRING BLOCK FIRST |
| 0750 | 22242 | 164307 | | JMP E15+1,I | P+1, NO LETTER FOUND |
| 0751 | 22243 | 162115 | | JSM VAR01,I | P+2, MUST BE NUMERIC OR ARRAY VARIABLE |
| 0752 | 22244 | 170402 | | RET | P+3, STRING VARIABLE FOUND AND STORED |
| 0753 | 22245 | 170402 | | RET | |
| 0754 | 22246 | 170402 | | RET | |
| 0755* | | | | | |
| 0756* | LOOK FOR THE STRING BLOCK AND SAVE IT'S ADDRESS | | | | |
| 0757* | | | | | |
| 0758 | 22247 | 020121 | STBLK | LDA B4000 | |
| 0759 | 22250 | 024130 | | LDB FROMA | |
| 0760 | 22251 | 074457 | | CPA B,I | |
| 0761 | 22252 | 066264 | | JMP FPAG2 | |
| 0762 | 22253 | 014131 | | CPB LROMA | |
| 0763 | 22254 | 066257 | | JMP FPAG1 | |

PAGE 450

EXTENDED I/O BLOCK'S CONSTANTS, SYNTAX, AND LINKAGES

0764 22255 004116
0765 22256 066251
0766 22257 025403 FPAG1
0767 22260 111403
0768 22261 066264
0769 22262 074742
0770 22263 066267
0771 22264 074442 FPAG2
0772 22265 074304
0773 22266 074070
0774 22267 035641
0775 22270 074117
0776 22271 170402

ADB QTOP
JMP *-5
LDB MAW
CPA MAW,1
JMP FPAG2
SBR 16
JMP *+4
SBR 10
SBL 10
SIB *+1
STB T5
LDA B
RET

PAGE 451

EXTENDED I/O BLOCK'S CONSTANTS, SYNTAX, AND LINKAGES

0778*

0779*

CMD STATEMENT SYNTAX

0780*

| | | | | | |
|------|-------|--------|------|-----|---------|
| 0781 | 22272 | 062247 | CMD5 | JSM | STBLK |
| 0782 | 22273 | 025641 | | LDB | T5 |
| 0783 | 22274 | 076150 | | RZB | *+3 |
| 0784 | 22275 | 061055 | | JSM | GNEXT |
| 0785 | 22276 | 066300 | | JMP | *+2 |
| 0786 | 22277 | 161641 | | JSM | T5,I |
| 0787 | 22300 | 045706 | | ISZ | SBPTR |
| 0788 | 22301 | 160245 | | JSM | GTSTA,I |
| 0789 | 22302 | 060504 | | JSM | COMCK |
| 0790 | 22303 | 065151 | | JMP | EOST |
| 0791 | 22304 | 066273 | | JMP | CMD5+1 |

0792*

0793*

0794*

CMD STATEMENT EXECUTION

0795*

| | | | | | |
|------|-------|--------|-----|-----|-----------------------|
| 0796 | 22305 | 062247 | CMD | JSM | STBLK |
| 0797 | 22306 | 022035 | | LDA | .SC13 |
| 0798 | 22307 | 031656 | | STA | T2 |
| 0799 | 22310 | 020013 | | LDA | B400 |
| 0800 | 22311 | 063342 | | JSM | OTPT |
| 0801 | 22312 | 162107 | | JSM | QTCHK,I QUOTE NEXT? |
| 0802 | 22313 | 062331 | | JSM | SVAR NO — TRY STRINGS |
| 0803 | 22314 | 063374 | | JSM | .OUTP |
| 0804 | 22315 | 020107 | | LDA | B1000 |
| 0805 | 22316 | 063342 | | JSM | OTPT |
| 0806 | 22317 | 021472 | | LDA | TEMPS |
| 0807 | 22320 | 011421 | | CPA | PRADD DONE WITH STMT? |
| 0808 | 22321 | 164172 | | JMP | XEC4A,I YES |
| 0809 | 22322 | 162107 | | JSM | QTCHK,I QUOTE NEXT? |
| 0810 | 22323 | 062331 | | JSM | SVAR NO — TRY STRINGS |
| 0811 | 22324 | 063374 | | JSM | .OUTP |
| 0812 | 22325 | 021472 | | LDA | TEMPS |
| 0813 | 22326 | 011421 | | CPA | PRADD DONE WITH STMT? |
| 0814 | 22327 | 164172 | | JMP | XEC4A,I |
| 0815 | 22330 | 066310 | | JMP | CMD+3 NO LOOP |

0816*

0817

0818

0819

0820

0821

0822

0823

0824*

0825*

0826*

FUNCTION EXECUTION

0827*

0828*

| | | | | | | |
|------|-------|--------|------|-----|------|----------------------------------|
| 0829 | 22340 | 062361 | EROT | JSM | SCHK | GO CHECK IF SYNTAX OR EXECUTION |
| 0830 | 22341 | 074076 | | TCB | | EXECUTION |
| 0831 | 22342 | 021644 | | LDA | T1 | LOAD THE ARGUMENT INTO (A) |
| 0832 | 22343 | 074153 | | SBP | *+3 | EXIT IF ARG WAS NEGATIVE OR ZERO |
| 0833 | 22344 | 074070 | | SIB | *+1 | |

PAGE 452 EXTENDED I/O BLOCK'S CONSTANTS, SYNTAX, AND LINKAGES

| | | | | | | |
|-------|-------|--------|-------|-----|---------|---------------------------------------|
| 0834 | 22345 | 070006 | | RAR | 1 | |
| 0835 | 22346 | 077770 | | RIB | *-1 | LOOP, ROTATING (A) ONE BIT UNTIL DONE |
| 0836 | 22347 | 066357 | | JMP | ESTO | EXIT LOOP |
| 0837* | | | | | | |
| 0838 | 22350 | 062361 | EIOR | JSM | SCHK | |
| 0839 | 22351 | 021644 | | LDA | T1 | T1 IOR T2 |
| 0840 | 22352 | 041656 | | IOR | T2 | |
| 0841 | 22353 | 066357 | | JMP | ESTO | |
| 0842* | | | | | | |
| 0843 | 22354 | 062361 | EIAND | JSM | SCHK | |
| 0844 | 22355 | 021644 | EAND | LDA | T1 | T1 AND T2 |
| 0845 | 22356 | 051656 | | AND | T2 | |
| 0846 | 22357 | 070137 | ESTO | LDB | A | |
| 0847 | 22360 | 164227 | | JMP | FXFLA,I | |
| 0848* | | | | | | |
| 0849 | 22361 | 021416 | SCHK | LDA | MODE | |
| 0850 | 22362 | 070552 | | SAM | .SCHK | SKIP IF SYNTAX PHASE |
| 0851* | | | | | | |
| 0852 | 22363 | 061217 | | JSM | FETCH | |
| 0853 | 22364 | 160225 | | JSM | FLTRA,I | |
| 0854 | 22365 | 024062 | | LDB | FLGBT | IF OVERFLOW USE 100000 OCTAL |
| 0855 | 22366 | 035644 | | STB | T1 | |
| 0856 | 22367 | 061217 | | JSM | FETCH | |
| 0857 | 22370 | 160225 | | JSM | FLTRA,I | |
| 0858 | 22371 | 164273 | | JMP | E5-1,I | |
| 0859 | 22372 | 035656 | | STB | T2 | |
| 0860 | 22373 | 045472 | | ISZ | TEMPS | |
| 0861 | 22374 | 170402 | | RET | | |
| 0862* | | | | | | |
| 0863* | | | | | | |
| 0864* | | | | | | |
| 0865 | 22375 | 055777 | SCHK | DSZ | RSPTR | |
| 0866 | 22376 | 121472 | | LDA | TEMPS,I | |
| 0867 | 22377 | 031641 | | STA | ST+1 | SAVE LEFT BRACKET COUNT |
| 0868 | 22400 | 061055 | | JSM | GNEXT | |
| 0869 | 22401 | 045706 | | ISZ | SBPTR | |
| 0870 | 22402 | 060521 | | JSM | LPCKE | |
| 0871 | 22403 | 160202 | | JSM | FSCA,I | |
| 0872 | 22404 | 060510 | | JSM | COMCE | |
| 0873 | 22405 | 160202 | | JSM | FSCA,I | |
| 0874 | 22406 | 025641 | | LDB | ST+1 | |
| 0875 | 22407 | 135472 | | STB | TEMPS,I | |
| 0876 | 22410 | 060534 | | JSM | RPCKE | |
| 0877 | 22411 | 064356 | | JMP | RETN3 | |
| 0878* | | | | | | |

PAGE 453 EXTENDED I/O BLOCK'S CONSTANTS, SYNTAX, AND LINKAGES

0880*
 0881* MISCELLANEOUS ROUTINES & ETC.
 0882*
 0883* SUBROUTINE TO RECORD CODE-CONVERSION TABLE
 0884*
 0885 22412 060510 RTBL JSM COMCE DEMAND COMMA
 0886 22413 061140 JSM LTR LETTER?
 0887 22414 164307 JMP E15+1,I NO, ERROR
 0888 22415 021474 LDA TEMPI RECORD ARRAY IDENTIFIER
 0889 22416 024020 LDB .3 OF UNKNOWN DIMENSIONS
 0890 22417 060451 JSM STROP+1
 0891 22420 021475 LDA TEMP2 RETRIEVE FOLLOWING CHARACTER
 0892 22421 170402 RET
 0893*
 0894* SUBROUTINE TO FIND A NON-FORMULA OPERATOR
 0895*
 0896 22422 026054 NFOP LDB NFOPA GET NON-FORMULA OPERATOR
 TABLE ADDRESS
 0897 22423 005642 ADB T8
 0898 22424 060546 JSM STEXP DO TABLE SEARCH
 0899 22425 170402 RET NOT FOUND
 0900 22426 060456 JSM SBPUD
 0901 22427 074742 SBR 16
 0902 22430 135706 STB SBPTR,I
 0903 22431 070202 SAR 5
 0904 22432 050040 AND .31
 0905 22433 064357 JMP RETN2
 0906*
 0907* RECORD I/O HEADING
 0908*
 0909 22434 160265 IOHD JSM IOHEA,I GET (S.C.,*
 0910 22435 060524 JSM RPCK RIGHT PAREN?
 0911 22436 066440 JMP *+2 NO
 0912 22437 065153 JMP EOLCK YES, IF EOL RETURN TO BASIC SYSTEM
 0913 22440 062412 JSM RTBL CHECK FOR CONVERSION TABLE
 0914 22441 060534 JSM RPCKE DEMAND RIGHT BRACKET
 0915 22442 065153 JMP EOLCK CHECK FOR EOL AND RETURN IF NOT



PAGE 454

FORMATTED INPUT BLOCK EXECUTION

```
0917*
0918* EVALUATE OCT EXPRESSION
0919*
0920 22443 060445 EOCT JSM XFAR2+1
0921 22444 074742 SBR 16
0922 22445 021754 LDA AR2 GET EXPONENT
0923 22446 070652 SAM *+13 AND TEST IT
0924 22447 000157 EQCI ADA M256
0925 22450 031466 STA LOCL1
0926 22451 074117 LDA B CHECK FOR POSSIBLE OVERFLOW
0927 22452 070602 SAR 13
0928 22453 072550 RZA *+11
0929 22454 171400 MLS GET NEXT DIGIT
0930 22455 074606 RBR 13
0931 22456 070037 ADB A ADD IN NEW DIGIT
0932 22457 050025 AND .8 TEST FOR LEGAL OCTAL DIGIT
0933 22460 072310 RZA *+6
0934 22461 021466 LDA LOCL1
0935 22462 071253 SAP EOCI
0936*
0937 22463 070111 SLA *+2 SKIP IF +
0938 22464 074076 TCB
0939 22465 164227 JMP FXFLA,I DONE
0940*
0941 22466 164273 JMP E5-1,I ERROR
0942*
0943 22467 060445 STAT JSM XFAR2+1
0944 22470 063461 JSM UNNM+1
0945 22471 040121 IOR B4000
0946 22472 172741 STF 1
0947 22473 172141 OTA 1
0948 22474 172741 STF 1
0949 22475 177241 LIB 1,C
0950 22476 074604 SBL 4
0951 22477 074542 SBR 12
0952 22500 164227 JMP FXFLA,I
0953*
0954 22501 060445 CHAR JSM XFAR2+1
0955 22502 063461 JSM UNNM+1
0956 22503 040121 IOR B4000
0957 22504 063447 JSM .CHAR
0958 22505 074073 SBP *+1,C
0959 22506 074302 SBR 7
0960 22507 164227 JMP FXFLA,I
0961*
0962 22510 062536 LIN JSM .LIN OUTPUT X LINE FEEDS AND CARRIAGE
RETURN
0963 22511 076150 RZB *+3
0964 22512 062546 JSM .LIN2
0965 22513 066533 JMP .LIN1 IF ARG=0, GIVE CARRIAGE RETURN
0966 22514 070112 SAM *+2
0967 22515 062546 JSM .LIN2 IF ARG IS NEGATIVE SKIP CARRIAGE
RETURN
0968 22516 020061 LDA LNFD
0969 22517 162111 JSM PRCHK,I
0970 22520 061353 JSM TTY
0971 22521 055644 DSZ TI
0972 22522 066516 JMP *-4
```

566

PAGE 455

FORMATTED INPUT BLOCK EXECUTION

```
0973 22523 066533      JMP .LIN1
0974*
0975 22524 062536 SPA   JSM .LIN      OUTPUT X SPACES
0976 22525 070312      SAM .LIN1
0977 22526 162103      JSM MOUTA,I
0978 22527 066533      JMP .LIN1
0979*
0980 22530 062536 BIN   JSM .LIN      OUTPUT THE BINARY CODE FOR X
0981 22531 052053      AND BIT12
0982 22532 061353      JSM TTY
0983 22533 020132 .LIN1 LDA M1
0984 22534 031755      STA AR2+1
0985 22535 170402      RET
0986*
0987 22536 060445 .LIN  JSM XFAR2+1
0988 22537 160226      JSM FLTFA,I
0989 22540 164273      JMP E5-1,I
0990 22541 074117      LDA B
0991 22542 074113      SBP *+2
0992 22543 074076      TCB
0993 22544 035644      STB T1
0994 22545 170402      RET
0995*
0996 22546 020031 .LIN2 LDA CRTN
0997 22547 162111      JSM PRCHK,I
0998 22550 061353      JSM TTY
0999 22551 070742      SAR 16
1000 22552 031622      STA TERM
1001 22553 170402      RET
```

561

554

1,444,141

554

PAGE 456

FORMATTED INPUT BLOCK EXECUTION

```

1003*
1004* EXECUTE GET STATEMENT
1005*
1006 22554 062247 EGET JSM STBLK CHECK IF STRING BLOCK IS IN.
1007 22555 045472 ISZ TEMPS
1008 22556 063460 JSM UNNM
1009 22557 040121 IOR B4000
1010 22560 031644 STA T1
1011 22561 022051 LDA FMTOP
1012 22562 031640 STA ST
1013 22563 074742 SBR 16
1014 22564 162117 JSM PLOTE,I
1015 22565 074742 SBR 16 P+1, NOT FOUND — FREE FIELD INPUT
1016 22566 035662 STB T9 P+2, FOUND FORMATTED INPUT
1017*
1018 22567 063471 JSM CTBL CHECK FOR CONVERSION TABLE
1019*
1020 22570 022034 LDA FLOP
1021 22571 031665 STA FLOOP INITIALIZE FOR LOOP POINTER
1022*
1023 22572 063073 JSM RREC READ AN INPUT RECORD
1024 22573 025662 LDB T9
1025 22574 076510 RZB EGE1
1026*
1027* FREE FIELD INPUT
1028*
1029 22575 062723 JSM NADD GET THE VARIABLE
1030 22576 061156 EGE1 JSM CONST FETCH A NUMBER
1031 22577 066603 JMP *+4 NO NUMBER FOUND, ZERO IT, AND THE
REMAINING NO
1032 22600 060412 JSM RESULT NO — STORE IT
1033 22601 066575 JMP EGE1-1 LOOP
1034 22602 062723 JSM NADD
1035 22603 061256 JSM CLAR2 SET THE REMAINING VARIABLES TO
ZERO
1036 22604 060412 JSM RESULT
1037 22605 066602 JMP *-3 LOOP
1038*
1039* FORMATTED INPUT
1040*
1041 22606 021456 EGE1 LDA FORMS
1042 22607 070073 SAP *+1,C
1043 22610 031456 STA FORMS
1044 22611 062723 EGE2 JSM NADD
1045 22612 162106 EGE3 JSM TYPEC,I GET A FORMAT SPEC.
1046 22613 066631 JMP ESLH+1 P+1, END OF FORMAT, READ NEXT
RECORD AND JUMP TO EGE
P+2, QUOTE FIELD NOT ALLOWED
1047 22614 066623 JMP E71
1048 22615 066642 JMP EFMT 3 F
1049 22616 066642 JMP EFMT 4 E
1050 22617 066623 JMP E71 5 $
1051 22620 066633 JMP BFMT 6 B
1052 22621 066625 JMP ESKP 7 X
1053 22622 066630 JMP ESLH 8 /
1054 22623 160205 E71 JSM AERRA,I INVALID FORMAT SPECIFICATION
1055 22624 177654 DEC -84
1056*

```

568

PAGE 457

FORMATTED INPUT BLOCK EXECUTION

| | | | | | | |
|-------|-------|--------|------|-----|---------|-----------------------|
| 1058 | 22625 | 162114 | ESKP | JSM | FSPR,I | |
| 1059 | 22626 | 063055 | | JSM | NBYT | |
| 1060 | 22627 | 066612 | | JMP | EGE3 | |
| 1061* | | | | | | |
| 1062 | 22630 | 162114 | ESLH | JSM | FSPR,I | |
| 1063 | 22631 | 063073 | | JSM | RREC | READ A NEW RECORD |
| 1064 | 22632 | 066612 | | JMP | EGE3 | |
| 1065* | | | | | | |
| 1066 | 22633 | 162114 | BFMT | JSM | FSPR,I | |
| 1067 | 22634 | 063055 | | JSM | NBYT | GET NEXT BYTE |
| 1068 | 22635 | 010061 | | CPA | LNFD | IF LINE FEED USE ZERO |
| 1069 | 22636 | 070742 | | SAR | 16 | |
| 1070 | 22637 | 070137 | | LDB | A | |
| 1071 | 22640 | 160227 | | JSM | FXFLA,I | CONVERT TO F.P. |
| 1072 | 22641 | 066670 | | JMP | EFM3+1 | |
| 1073* | | | | | | |
| 1074 | 22642 | 045466 | EFMT | ISZ | LOCL1 | |
| 1075 | 22643 | 121466 | | LDA | LOCL1,I | GET W SPEC |
| 1076 | 22644 | 045466 | | ISZ | LOCL1 | |
| 1077 | 22645 | 045466 | | ISZ | LOCL1 | |
| 1078 | 22646 | 070076 | | TCA | | |
| 1079 | 22647 | 031647 | | STA | TW | |
| 1080 | 22650 | 125466 | | LDB | LOCL1,I | GET D SPEC |
| 1081 | 22651 | 074017 | | ADA | B | |
| 1082 | 22652 | 070056 | | CMA | | |
| 1083 | 22653 | 070404 | | SAL | 8 | |
| 1084 | 22654 | 031651 | | STA | TD | |
| 1085 | 22655 | 162114 | | JSM | FSPR,I | |
| 1086 | 22656 | 021647 | | LDA | TW | SET FIELD WIDTH |
| 1087 | 22657 | 031650 | | STA | TX | |
| 1088 | 22660 | 070742 | | SAR | 16 | CLEAR IGNORE-D FLAG |
| 1089 | 22661 | 031662 | | STA | T9 | |
| 1090 | 22662 | 063153 | | JSM | NMBR | CALL FOR REAL NUMBER |
| 1091 | 22663 | 021662 | | LDA | T9 | ANY .ORE? |
| 1092 | 22664 | 072150 | | RZA | EFM3 | YES |
| 1093 | 22665 | 021651 | | LDA | TD | NO, ADJUST EXPONENT |
| 1094 | 22666 | 031754 | | STA | AR2 | |
| 1095 | 22667 | 061254 | EFM3 | JSM | ZONK | |
| 1096 | 22670 | 060412 | | JSM | RESULT | |
| 1097 | 22671 | 066611 | | JMP | EGE2 | |
| 1098* | | | | | | |



556

1,444,141

556

PAGE 458

FORMATTED INPUT BLOCK EXECUTION

```
1100*
1101 22672 050040 .NADD AND B37 FIND NUMERIC OR STRING VARIABLE
1102 22673 010076 CPA B36 STRING NEXT?
1103 22674 066676 JMP *+2 YES
1104 22675 164201 JMP FORM,I NO, GET THE VARIABLE'S ADDRESS
1105 22676 021641 LDA T5
1106 22677 072110 RZA *+2
1107 22700 164311 JMP E15+3,I STRINGS NOT AVAILABLE
1108 22701 000012 ADA .1 JSM GADD ON THE STRING BLOCK
1109 22702 070717 JSM A,I
1110 22703 021400 LDA BADDR
1111 22704 031645 STA PTR1 PTR1=CURRENT WORD OF BUFFER
1112 22705 070076 TCA
1113 22706 001623 ADA PEND
1114 22707 031647 STA TMPRE TMPRE=REMAINING LENGTH OF THE
                                         BUFFER
1115 22710 024132 LDB M1
1116 22711 020017 LDA .2 JSM PSTR (S.B.), PREPARE DESTINATION
1117 22712 001641 ADA T5
1118 22713 070717 JSM A,I
1119 22714 021654 LDA TNULL
1120 22715 070056 CMA
1121 22716 001400 ADA BADDR
1122 22717 031400 STA BADDR
1123 22720 020020 LDA .3 JSM TRSTR(S.B.), TRANSFER DATA
1124 22721 001641 ADA T5
1125 22722 070717 JSM A,I
1126*
1127* TRANSFER COMPLETE GO GET NEXT VARIABLE
1128*
1129*
1130* SUBROUTINE TO GET AN ADDRESS
1131*
1132 22723 021472 .NADD LDA TEMPS
1133 22724 011421 CPA PRADD DONE?
1134 22725 067443 JMP FXIT
1135 22726 121472 .NADI LDA TEMPS,I
1136 22727 010121 CPA B4000 " , " ?
1137 22730 066742 JMP PFOR GO SET UP FOR LOOP
1138 22731 010004 CPA B10K RIGHT PAREN.?
1139 22732 066734 JMP *+2
1140 22733 066672 JMP .NADD
1141*
1142 22734 021472 LDA TEMPS IS IT
1143 22735 070070 SIA *+1 FOLLOWED BY
1144 22736 070517 LDA A,I
1145 22737 010126 CPA LPOP ( LEFT PAREN.?
1146 22740 066742 JMP PFOR YES, GO SET FOR LOOP
1147 22741 067003 JMP TFOR NO, MUST BE DURING EXECUTION OF
                                         FOR LOOP
1148*
1149 22742 021665 PFOR LDA FLOOP
1150 22743 000022 ADA .5 BUMP FOR LOOP POINTER
1151 22744 031665 STA FLOOP
1152 22745 021472 LDA TEMPS
1153 22746 000020 ADA .3
1154 22747 031472 STA TEMPS MOVE TEMPS PASS "FOR"
1155 22750 160201 JSM FORM,I EVALUATE STARTING VALUE
```

PAGE 459

FORMATTED INPUT BLOCK EXECUTION

| | | | | | | |
|-------|-------|--------|------|-----|---------|---|
| 1156 | 22751 | 021413 | | LDA | HSTPT | |
| 1157 | 22752 | 000133 | | ADA | M2 | |
| 1158 | 22753 | 025665 | | LDB | FLOOP | |
| 1159 | 22754 | 170004 | | XFR | | MOVE THE VARIABLE NAME AND ADDRESS TO |
| 1160 | 22755 | 004132 | | ADB | M1 | THE LOOP BLOCK |
| 1161 | 22756 | 035663 | | STB | .FLP | SET .FLP TO FLOOP+2 |
| 1162 | 22757 | 045472 | | ISZ | TEMPS | "TO" |
| 1163 | 22760 | 061217 | | JSM | FETCH | EVALUATE FINAL VALUE |
| 1164 | 22761 | 160225 | | JSM | FLTRA,I | |
| 1165 | 22762 | 164273 | | JMP | E5-1,I | |
| 1166 | 22763 | 074076 | | TCB | | |
| 1167 | 22764 | 135663 | | STB | .FLP,I | SAVE "FINAL" |
| 1168 | 22765 | 045663 | | ISZ | .FLP | |
| 1169 | 22766 | 024012 | | LDB | .I | |
| 1170 | 22767 | 121472 | | LDA | TEMPS,I | |
| 1171 | 22770 | 002050 | | ADA | STEC | |
| 1172 | 22771 | 072250 | | RZA | *+5 | ANY STEP ? |
| 1173 | 22772 | 045472 | | ISZ | TEMPS | YES, MOVE PASS "STEP" |
| 1174 | 22773 | 061217 | | JSM | FETCH | |
| 1175 | 22774 | 160225 | | JSM | FLTRA,I | |
| 1176 | 22775 | 164273 | | JMP | E5-1,I | |
| 1177 | 22776 | 135663 | | STB | .FLP,I | SAVE STEP |
| 1178 | 22777 | 045663 | | ISZ | .FLP | |
| 1179 | 23000 | 025472 | | LDB | TEMPS | |
| 1180 | 23001 | 135663 | | STB | .FLP,I | SAVE THE ADDRESS AFTER FOR LOOP INFORMATION |
| 1181 | 23002 | 066726 | | JMP | NAD1 | |
| 1182* | | | | | | |
| 1183* | | | | | | |
| 1184 | 23003 | 025665 | TFOR | LDB | FLOOP | |
| 1185 | 23004 | 074070 | | SIB | *+1 | |
| 1186 | 23005 | 061372 | | JSM | FIX+1 | |
| 1187 | 23006 | 031660 | | STA | T6 | |
| 1188 | 23007 | 010025 | | CPA | .8 | WAS IT AN INTEGER? |
| 1189 | 23010 | 067015 | | JMP | *+5 | YES |
| 1190 | 23011 | 162113 | | JSM | OPCH1,I | |
| 1191 | 23012 | 074117 | | LDA | B | |
| 1192 | 23013 | 060445 | | JSM | XFAR2+1 | |
| 1193 | 23014 | 160225 | | JSM | FLTRA,I | |
| 1194 | 23015 | 074537 | | LDB | B,I | FOR VARIABLE IN (B), NO OVERFLOW |
| 1195 | 23016 | 021665 | | LDA | FLOOP | |
| 1196 | 23017 | 000020 | | ADA | .3 | |
| 1197 | 23020 | 070437 | | ADB | A,I | ADD THE STEP SIZE TO B |
| 1198 | 23021 | 021660 | | LDA | T6 | |
| 1199 | 23022 | 002036 | | ADA | M8 | |
| 1200 | 23023 | 072210 | | RZA | *+4 | IF NOT AN INTEGER SKIP 4 |
| 1201 | 23024 | 121665 | | LDA | FLOOP,I | YES IT WAS AN INTEGER |
| 1202 | 23025 | 070577 | | STB | A,I | |
| 1203 | 23026 | 067036 | | JMP | TFOR | |
| 1204 | 23027 | 035660 | | STB | T6 | |
| 1205 | 23030 | 160227 | | JSM | FXFLA,I | |
| 1206 | 23031 | 025665 | | LDB | FLOOP | |
| 1207 | 23032 | 074070 | | SIB | *+1 | |
| 1208 | 23033 | 061372 | | JSM | FIX+1 | |
| 1209 | 23034 | 060416 | | JSM | STORE | |
| 1210 | 23035 | 025660 | | LDB | T6 | |
| 1211 | 23036 | 021665 | TFOR | LDA | FLOOP | |

PAGE 460

FORMATTED INPUT BLOCK EXECUTION

| | | | | | |
|-------|-------|--------|-----|-------|--------------------------------|
| 1212 | 23037 | 000017 | ADA | .2 | |
| 1213 | 23040 | 070437 | ADB | A,I | CHECK IF AT FINAL VALUE |
| 1214 | 23041 | 074110 | SZB | *+2 | |
| 1215 | 23042 | 074313 | SBP | *+6 | |
| 1216 | 23043 | 021665 | LDA | FLOOP | NO |
| 1217 | 23044 | 000021 | ADA | .4 | |
| 1218 | 23045 | 070517 | EDA | A,I | |
| 1219 | 23046 | 031472 | STA | TEMPS | |
| 1220 | 23047 | 066726 | JMP | NADI | |
| 1221* | | | | | |
| 1222 | 23050 | 045472 | ISZ | TEMPS | |
| 1223 | 23051 | 021665 | LDA | FLOOP | |
| 1224 | 23052 | 000136 | ADA | M5 | |
| 1225 | 23053 | 031665 | STA | FLOOP | DONE WITH LOOP BUMP TEMPS, AND |
| | | | | | LOOP POINTER |
| 1226 | 23054 | 066723 | JMP | NADD | |
| 1227* | | | | | |

PAGE 461

FORMATTED INPUT BLOCK EXECUTION

| | | | | | |
|-------|-----------------------------|--------|------|---------|---------------------------|
| 1229* | | | | | |
| 1230* | SUBROUTINE #1 TO GET A BYTE | | | | |
| 1231* | | | | | |
| 1232 | 23055 | 061040 | NBYT | JSM | GETCH-1 GET A CHARACTER |
| 1233 | 23056 | 010061 | CPA | LNFD | |
| 1234 | 23057 | 155466 | DSZ | LOCL1,I | |
| 1235 | 23060 | 170402 | RET | | |
| 1236* | | | | | |
| 1237* | SUBROUTINE #2 TO GET A BYTE | | | | |
| 1238* | | | | | |
| 1239 | 23061 | 021650 | NBYB | LDA | TX |
| 1240 | 23062 | 072170 | | RIA | *+3 |
| 1241 | 23063 | 020061 | | LDA | LNFD |
| 1242 | 23064 | 170402 | | RET | RETURN LINEFEED IF AT END |
| 1243* | | | | | |
| 1244 | 23065 | 031650 | | STA | TX |
| 1245 | 23066 | 063055 | | JSM | NBYT |
| 1246 | 23067 | 010041 | | CPA | 32 |
| 1247 | 23070 | 067061 | | JMP | NBYB |
| 1248 | 23071 | 170402 | | RET | IGNORE BLANKS |

PAGE 462

FORMATTED INPUT BLOCK EXECUTION

1250 23072 164271 MEOV JMP EI+1,1 MEMORY OVERFLOW
 1251*
 1252* SUBROUTINE TO READ ONE ASCII RECORD INTO BUFFER
 1253*
 1254 23073 021412 RREC LDA LSTPT
 1255 23074 000061 ADA .10 ALLOW 10 WORDS IN LOW-STACK FOR
 FORMT
 1256 23075 070744 SAL 1
 1257 23076 031400 STA BADDR
 1258 23077 031706 STA SBPTR
 1259 23100 002046 ADA .80
 1260 23101 031623 STA PEND
 1261 23102 070002 SAR 1
 1262 23103 070076 TCA
 1263 23104 001413 ADA HSTPT
 1264 23105 071252 SAM MEOV
 1265*
 1266 23106 020141 LDA M10 SET FOR 10 BLANKS
 1267 23107 031466 STA LOCL1 THE CARD READER ONLY GIVES 10
 1268 23110 063321 JSM INPT
 1269 23111 072310 RZA RRE1+1 SKIP IF NOT A BLANK
 1270 23112 045466 ISZ LOCL1
 1271 23113 067110 JMP *-3
 1272*
 1273 23114 160205 JSM AERRA,1 ERROR 83 — END OF TAPE
 1274 23115 177655 DEC -83
 1275*
 1276*
 1277 23116 063321 RRE1 JSM INPT CALL FOR INPUT
 1278 23117 010061 CPA LNFD IS IT LINEFEED?
 1279 23120 067131 JMP BFND YES STORE IN BUFFER AND RETURN
 1280 23121 012041 CPA AROW IS IT LEFT-ARROW?
 1281 23122 067142 JMP BKUP YES, BACK UP
 1282 23123 012042 CPA ALTM IS IT ALT MODE?
 1283 23124 067147 JMP SKND YES, RE-READ RECORD
 1284 23125 012043 CPA ESCM IS IT ESC MODE?
 1285 23126 067147 JMP SKND YES, RE-READ RECORD
 1286 23127 061016 JSM ONEXT NO, STORE IN BUFFER
 1287 23130 067116 JMP RRE1
 1288*
 1289 23131 045623 BFND ISZ PEND MOVE END POINTER
 1290 23132 020031 LDA EOL
 1291 23133 061016 JSM ONEXT
 1292 23134 021400 LDA BADDR
 1293 23135 000132 ADA M1 SET PEND TO THE EOL CHAR.
 1294 23136 031623 STA PEND
 1295 23137 021706 LDA SBPTR
 1296 23140 031400 STA BADDR
 1297 23141 170402 RET
 1298*
 1299 23142 021400 BKUP LDA BADDR
 1300 23143 011706 CPA SBPTR
 1301 23144 067116 JMP RRE1 DON'T GO BEYOND START OF BUFFER
 1302 23145 055400 DSZ BADDR BACK UP
 1303 23146 067116 JMP RRE1
 1304*
 1305 23147 063321 SKND JSM INPT SKIP TO END OF RECORD

560

560

1,444,141

560

PAGE 463

FORMATTED INPUT BLOCK EXECUTION

1306 23150 010061
1307 23151 067073
1308 23152 067147
1309*

CPA LNFD
JMP RREC RE-READ
JMP *-3

PAGE 464

FORMATTED INPUT BLOCK EXECUTION

```

1311*
1312* SUBROUTINE TO BUILD A NUMBER IN AR2
1313*
1314 23153 061256 NMBR JSM CLAR2 SET AR2 = 0.
1315 23154 022040 LDA AMF
1316 23155 170000 CLR CLEAR WORK AREA
1317 23156 020132 LDA M1
1318 23157 031660 STA T6 M = - 1
1319 23160 020030 LDA .12
1320 23161 031661 STA T7 N = 12
1321*
1322 23162 063061 JSM NBYB GET FIRST CHARACTER
1323 23163 024132 LDB M1
1324 23164 010050 CPA ASCM IS IT -?
1325 23165 074742 SBR 16 YES
1326 23166 035712 STB .SIGN
1327 23167 010050 CPA ASCM BYPASS-OR+
1328 23170 067172 JMP *+2
1329 23171 010047 CPA ASCP
1330*
1331 23172 063061 NMB1 JSM NBYB GET NEXT CHARACTER
1332 23173 060750 JSM DIGCK IS IT A DIGIT?
1333 23174 067222 JMP NMB3 NO
1334 23175 045652 ISZ MF YES
1335 23176 076350 RZB NMB2 IS IT ZERO?
1336 23177 021653 LDA SD YES, ANY SIGNIFICANT DIGITS YET?
1337 23200 072310 RZA NMB2+1 YES
1338 23201 021654 LDA DP NO, ANY DECIMAL POINT YET?
1339 23202 071410 SZB NMB1 NO, IGNORE THIS ZERO
1340 23203 055660 DSZ T6 YES, M=M-1
1341 23204 067172 JMP NMB1
1342*
1343 23205 045653 NMB2 ISZ SD SIGNIFICANT DIGIT (1-9)
1344 23206 021654 LDA DP
1345 23207 072150 RZA *+3 IS DP=0?
1346 23210 045660 ISZ T6 YES, M=M+1
1347 23211 067212 JMP *+1 (NOP)
1348 23212 021661 LDA T7 IS N=0?
1349 23213 070310 SZB *+6 YES, IGNORE REMAINING DIGITS
1350 23214 055661 DSZ T7 NO, N=N-1
1351 23215 067216 JMP *+1 (NOP)
1352 23216 171400 MLS
1353 23217 005757 ADB AR2+3 INCLUDE DIGIT IN AR2
1354 23220 035757 STB AR2+3
1355 23221 067172 JMP NMB1
1356*
1357 23222 010051 NMB3 CPA ASC. DECIMAL POINT?
1358 23223 067251 JMP NMB5 YES
1359 23224 012044 CPA ASCE EEX?
1360 23225 067256 JMP NMB6 YES
1361 23226 021652 LDA MF NO, IS MF=0?
1362 23227 072150 RZA *+3 IF MF IS ZERO, AND THERE WAS A (-)
SIGN - ERROR
1363 23230 025712 LDB .SIGN
1364 23231 074710 SZB E73
1365 23232 021660 LDA T6 NO, TEST FOR EXPONENT OVERFLOW
1366 23233 070137 NMB4 LDB A

```

575

562

1,444,141

562

PAGE 465

FORMATTED INPUT BLOCK EXECUTION

| | | | | | | |
|-------|-------|--------|------|-----|---------|------------------------------------|
| 1367 | 23234 | 074113 | | SBP | *+2 | |
| 1368 | 23235 | 074076 | | TCB | | |
| 1369 | 23236 | 004156 | | ADB | M100 | |
| 1370 | 23237 | 074413 | | SBP | E73 | OVERFLOW ERROR |
| 1371 | 23240 | 171450 | | NRM | | |
| 1372 | 23241 | 070254 | | SES | *+5 | SKIP IF ZERO MANTISSA |
| 1373 | 23242 | 070404 | | SAL | 8 | |
| 1374 | 23243 | 045712 | | ISZ | .SIGN | TEST MANTISSA SIGN (-1=*) |
| 1375 | 23244 | 070070 | | SIA | *+1 | |
| 1376 | 23245 | 031754 | | STA | AR2 | |
| 1377 | 23246 | 170402 | | RET | | |
| 1378* | | | | | | |
| 1379 | 23247 | 160205 | E73 | JSM | AERRA,1 | NUMBER SYNTAX ERROR |
| 1380 | 23250 | 177652 | | DEC | -86 | |
| 1381 | 23251 | 031662 | NMB5 | STA | T9 | SET IGNORE—D FLAG |
| 1382 | 23252 | 021654 | | LDA | DP | TEST DP FLAG |
| 1383 | 23253 | 073610 | | RZA | E73 | SET ALREADY, ERROR |
| 1384 | 23254 | 045654 | | ISZ | DP | |
| 1385 | 23255 | 067172 | | JMP | NMB1 | |
| 1386* | | | | | | |
| 1387 | 23256 | 031662 | NMB6 | STA | T9 | SET IGNORE—D FLAG |
| 1388 | 23257 | 021652 | | LDA | MF | |
| 1389 | 23260 | 072250 | | RZA | *+5 | MANTISSA DIGITS ENTERED |
| 1390 | 23261 | 021654 | | LDA | DP | ANY DECIMAL POINT? |
| 1391 | 23262 | 073250 | | RZA | E73 | YES, BUT NO MANTISSA DIGITS, ERROR |
| 1392 | 23263 | 020020 | | LDA | ONE | NO, SET AR2=1. |
| 1393 | 23264 | 060445 | | JSM | XFAR2+1 | |
| 1394 | 23265 | 063061 | | JSM | NBYB | GET NEXT CHARACTER |
| 1395 | 23266 | 024132 | | LDB | M1 | |
| 1396 | 23267 | 010050 | | CPA | ASCM | IS IT -? |
| 1397 | 23270 | 074742 | | SBR | 16 | YES |
| 1398 | 23271 | 035654 | | STB | DP | USE DP FOR EXPONENT SIGN |
| 1399 | 23272 | 010050 | | CPA | ASCM | BYPASS - OR + |
| 1400 | 23273 | 067275 | | JMP | *+2 | |
| 1401 | 23274 | 010047 | | CPA | ASCP | |
| 1402 | 23275 | 063061 | | JSM | NBYB | GET NEXT CHARACTER |
| 1403 | 23276 | 060750 | | JSM | DIGCK | DIGIT? |
| 1404 | 23277 | 067247 | | JMP | E73 | NO, ERROR |
| 1405 | 23300 | 035655 | NMB7 | STB | EX | YES |
| 1406 | 23301 | 063061 | | JSM | NBYB | GET NEXT CHARACTER |
| 1407 | 23302 | 060750 | | JSM | DIGCK | DIGIT? |
| 1408 | 23303 | 067313 | | JMP | NMB8 | NO |
| 1409 | 23304 | 021655 | | LDA | EX | YES |
| 1410 | 22305 | 070744 | | SAL | 1 | |
| 1411 | 23306 | 031466 | | STA | LOCL1 | |
| 1412 | 23307 | 070704 | | SAL | 2 | |
| 1413 | 23310 | 001466 | | ADA | LOCL1 | |
| 1414 | 23311 | 070037 | | ADB | A | |
| 1415 | 23312 | 067300 | | JMP | NMB7 | |
| 1416* | | | | | | |
| 1417 | 23313 | 021655 | NMB8 | LDA | EX | |
| 1418 | 23314 | 025654 | | LDB | DP | |
| 1419 | 23315 | 076110 | | RZB | *+2 | ADJUST EXPONENT SIGN |
| 1420 | 23316 | 070076 | | TCA | | |
| 1421 | 23317 | 001660 | | ADA | T6 | |
| 1422 | 23320 | 067233 | | JMP | NMB4 | |

PAGE 466

FORMATTED INPUT BLOCK EXECUTION

1424*
 1425* INPUT DRIVER
 1426*
 1427* T1 HAS SELECT CODE
 1428*
 1429 23321 021644 INPT LDA T1
 1430 23322 063447 JSM .CHAR GET A CHAR IN B
 1431 23323 074117 LDA B
 1432 23324 070073 SAP *+1,C
 1433 23325 070302 SAR 7
 1434 23326 025642 LDB T8 MASK DATA
 1435 23327 070510 SZA RETI CONVERSION NEEDED?
 1436 23330 074310 SZB *+6 IF(A)=0, DON'T CONVERT
 1437 23331 000132 ADA M1 IF(B) IS 0, SKIP CONVERSION
 1438 23332 070037 ADB A YES, TREAT INPUT CODE AS
 ADDRESS
 1439 23333 074517 LDA B,I
 1440 23334 070113 SAP *+2
 1441 23335 067505 JMP E79
 1442 23336 050104 AND B177
 1443 23337 010031 CPA CRTN
 1444 23340 067321 JMP INPT
 1445 23341 170402 RETI RET IGNORE CARRIAGE RETURNS
 RETURN DATA IN A—REGISTER
 1446*
 1447* OUTPUT DRIVER
 1448*
 1449* T2 HAS SELECT CODE
 1450*
 1451* LDA DATA
 1452* JSM OTPT
 1453*
 1454 23342 031466 OTPT STA LOCL1
 1455 23343 021656 LDA T2
 1456 23344 063431 JSM IOST CHECK I/O STATUS
 1457 23345 041466 IOR LOCL1 GET DATA
 1458 23346 065356 JMP TTY+3 STF 1, OTA 1, STC 1, AND RETURN
 1459*
 1460 23347 000143 LINEF DEF .10+.10+1
 1461 23350 000063 CARRF DEF CRTN+CRTN+1
 1462*
 1463 23351 027350 .CRLF LDB CARRF CARRIAGE RETURN
 1464 23352 063354 JSM *+2
 1465 23353 027347 LDB LINEF LINE FEED
 1466 23354 020012 LDA .1
 1467 23355 067374 JMP .OUTP
 1468*
 1469 23356 025642 .OUT. LDB T8
 1470 23357 074510 SZB OUTI
 1471 23360 074457 CPA B,I
 1472 23361 067365 JMP *+4
 1473 23362 015643 CPB TBEND
 1474 23363 170402 RET IF NO MATCH FOUND— ASSUME A
 NULL CHARACTER
 1475 23364 077630 RIB *-4
 1476 23365 021642 LDA T8
 1477 23366 070076 TCA
 1478 23367 074017 ADA 8
 1479 23370 070070 SIA *+1

564

1,444,141

564

PAGE 467

FORMATTED INPUT BLOCK EXECUTION

| | | | | | | |
|-------|-------|--------|-------|-----|-------|---|
| 1480 | 23371 | 065353 | OUTI | JMP | TTY | |
| 1481* | | | | | | |
| 1482 | 23372 | 025706 | | LDB | SBPTR | |
| 1483 | 23373 | 060374 | | JSM | WBUFF | |
| 1484 | 23374 | 031741 | .OUTP | STA | XC+1 | |
| 1485 | 23375 | 035740 | | STB | XC | |
| 1486 | 23376 | 025641 | | LDB | T5 | |
| 1487 | 23377 | 076350 | | RXB | XFTOS | ANY INTERNAL CONVERSION |
| 1488 | 23400 | 070250 | | SZA | *+5 | |
| 1489 | 23401 | 061053 | | JSM | GTMP | |
| 1490 | 23402 | 063356 | .OUTT | JSM | .OUT. | NO — OUTPUT THE DATA TO THE
DEVICE |
| 1491 | 23403 | 055741 | | DSZ | XC+1 | |
| 1492 | 23404 | 067401 | | JMP | *-3 | |
| 1493 | 23405 | 170402 | | RET | | |
| 1494* | | | | | | |
| 1495 | 23406 | 031647 | XFTOS | STA | TMPRE | |
| 1496 | 23407 | 025740 | | LDB | XC | |
| 1497 | 23410 | 035645 | | STB | PTR1 | PLACE THE COUNT AND ADDRESS IN
TMPRE, PTR1 |
| 1498 | 23411 | 022034 | | LDA | FLOP | |
| 1499 | 23412 | 025412 | | LDB | LSTPT | TRANSFER THE SUBSCRIPTS TO THE
LSTPT |
| 1500 | 23413 | 074070 | | SIB | *+1 | |
| 1501 | 23414 | 170004 | | XFR | | |
| 1502 | 23415 | 004132 | | ADB | M1 | |
| 1503 | 23416 | 035412 | | STB | LSTPT | BUMP THE LOW STACK POINTER |
| 1504 | 23417 | 021647 | | LDA | TMPRE | |
| 1505 | 23420 | 001662 | | ADA | T9 | |
| 1506 | 23421 | 031662 | | STA | T9 | |
| 1507 | 23422 | 021641 | | LDA | T5 | |
| 1508 | 23423 | 000017 | | ADA | .2 | JSM PSTR |
| 1509 | 23424 | 024132 | | LDB | M1 | |
| 1510 | 23425 | 070717 | | JSM | A,I | PREPARE THE DESTINATION |
| 1511 | 23426 | 021641 | | LDA | T5 | |
| 1512 | 23427 | 000020 | | ADA | .3 | JSM TRSTR, TRANSFER DATA |
| 1513 | 23430 | 070737 | | JMP | A,I | |

PAGE 468

FORMATTED INPUT BLOCK EXECUTION

1515*
 1516* CHECK I/O STATUS
 1517*
 1518* LDA SELECT CODE (SAVED)
 1519* JSM IOST
 1520*
 1521 23431 025442 IOST LDB STPFL CHECK STOP FLAG
 1522 23432 014040 CPB .31
 1523 23433 067443 JMP FXIT
 1524 23434 172741 STF 1 TURN OFF INTERRUPT SYSTEM
 1525 23435 172141 OTA 1 OUTPUT SELECT CODE
 1526 23436 172741 STF 1 LOAD DEVICE STATUS
 1527 23437 177241 LIB 1,C AND TURN ON INTERRUPT
 1528 23440 074406 RBR 9
 1529 23441 075413 SBP IOST LOOP IF BUSY
 1530 23442 170402 RET
 1531*
 1532* EXIT ROUTINE
 1533*
 1534 23443 070742 FXIT SAR 16
 1535 23444 031455 STA FORMAT
 1536 23445 160170 JSM CLERA,I
 1537 23446 164172 JMP XEC4A,I
 1538*
 1539 23447 012037 CHAR CPA SC13 SELECT CODE 13 ?
 1540 23450 067452 JMP *+2 YES, SKIP
 1541 23451 063456 JSM CEO-1 NO, SET CONTROL FIRST
 1542 23452 063431 JSM IOST GET A BYTE
 1543 23453 012037 CPA SC13 SELECT CODE 13 ?
 1544 23454 065356 JMP TTY+3 YES, SET CONTROL AFTER
 1545 23455 170402 RET
 1546*
 1547 23456 063431 JSM IOST
 1548 23457 065356 CEO JMP TTY+3 SET CONTROL

579

566

1,444,141

566

PAGE 469

FORMATTED INPUT BLOCK EXECUTION

```

1550*
1551* SAVE STACK POINTERS
1552* EVALUATE AND CHECK UNIT #
1553*
1554 23460 061217 UNNM JSM FETCH
1555 23461 160225 JSM FLTRA,I
1556 23462 164342 JMP E40+3,I
1557 23463 074117 LDA B
1558 23464 075710 SZB *-2 ERROR, ZERO SELECT CODE
1559 23465 074142 SBR 4
1560 23466 077610 RZB *-4 ERROR, SELECT CODE >15 OR <0
1561 23467 070146 RAR 4
1562 23470 170402 RET
1563*
1564* GET OPTIONAL CONVERSION TABLE
1565*
1566 23471 121472 CTBL LDA TEMPS,I
1567 23472 050065 AND OPMSK
1568 23473 010004 CPA RPOP ANY CONVERSION TABLE
1569 23474 067514 JMP CTB1 NO, STORE ZERO
1570 23475 121472 LDA TEMPS,I YES, GET ARRAY NAME
1571 23476 050115 AND B1777
1572 23477 160177 JSM SSYMA,I SEARCH TABLE
1573 23500 067505 JMP E79 ERROR, NOT FOUND
1574 23501 074517 LDA B,I GET TYPE INFORMATION
1575 23502 050037 AND .28
1576 23503 010025 CPA .8 INTEGER?
1577 23504 067507 JMP *+3
1578*
1579 23505 160205 E79 JSM AERRA,I NO SUCH TABLE OR UNSUITABLE
TABLE
1580 23506 177651 DEC -87
1581*
1582 23507 004132 ADB M1
1583 23510 074517 LDA B,I GET BASE ADDRESS OF TABLE
1584 23511 063515 JSM CTB1+1 GO COMPUTE LAST ADDRESS
1585 23512 045472 ISZ TEMPS
1586 23513 170402 RET
1587*
1588 23514 070742 CTBI SAR 16
1589 23515 031642 STA T8
1590 23516 071650 SZA CTB1-1 RETURN IF NOT CONVERSION
1591 23517 000132 ADA M1
1592 23520 070517 LDA A,I
1593 23521 070137 LDB A GET ROW AND COLUMN INFORMATION
1594 23522 074342 SBR 8
1595 23523 074070 SIB *+1 (B) = # OF ROW
1596 23524 050105 AND B377
1597 23525 070070 SIA *+1 (A) = # OF COLUMNS
1598 23526 061201 JSM IMPY
1599 23527 001642 ADA T8
1600 23530 031643 STA TBEND SAVE THE END OF TABLE ADDRESS
1601 23531 170402 RET

```

PAGE 470

FORMATTED INPUT BLOCK EXECUTION

1603*
1604* PREPARE FOR INTERNAL CONVERSION
1605*
1606 23532 072110 INTCV RZA *+2
1607 23533 164311 JMP E15+3,I STRING BLOCK NOT AVAILABE
1608 23534 000012 ADA .1
1609 23535 070717 JSM A,I JSM GADD , PROCESS POINTERS TO
DESTINATIONS
1610 23536 021412 LDA LSTPT GADD SET'S UP STRING POINTERS ON
LOW STACK.
1611 23537 000133 ADA M2 THEY MUST BE SAVED.
1612 23540 026034 LDB FLOP
1613 23541 170004 XFR
1614 23542 000135 ADA M4
1615 23543 031412 STA LSTPT
1616 23544 022051 LDA FMTOP
1617 23545 031640 STA ST
1618 23546 074742 SBR 16
1619 23547 162117 JSM PLOTE,I LOOK FOR A FORMAT STATEMENT
1620 23550 067722 JMP EWRI1-1 P+1, NOT FOUND, USE PRINT EXECUTION
1621 23551 067567 JMP .EOUT+1 P+2, FOUND
1622*
1623* OUTPUT STATEMENT EXECUTION
1624*
1625 23552 062247 EOUT JSM STBLK LOOK FOR THE STRING BLOCK
1626 23553 045472 ISZ TEMPS
1627 23554 125472 LDB TEMPS,I
1628 23555 074244 SBL 11
1629 23556 074542 SBR 12
1630 23557 014032 CPB .15 STRING ARRAY ROOM?
1631 23560 067532 JMP INTCV YES
1632 23561 055472 DSZ TEMPS NO
1633 23562 074742 SBR 16
1634 23563 035641 STB T5
1635 23564 022051 LDA FMTOP
1636 23565 162105 JSM SFMT,I SET UP DEVICE AND FORMAT CODES
1637 23566 067722 .EOUT JMP EWRI1-1 NO FORMAT SPECIFIED
1638 23567 063471 JSM CTBL CHECK CONVERSION TABLE
1639 23570 162112 JSM CKEOL,I NULL LIST?
1640 23571 000103 BLNKA DEF .32+.32+1 IMPOSSIBLE STATE — WORD USED AS A
CONSTANT
1641 23572 035472 STB TEMPS YES, SET TEMPS=PRADD
1642 23573 162106 WRIT2 JSM TYPEC,I GET NEXT FORMAT SPEC
1643 23574 067606 JMP WRITN END OF FORMAT
1644 23575 067615 JMP WRITQ QUOTE FIELD
1645 23576 074742 SBR 16 SET FLAG FOR FIXED
1646 23577 060563 JSM SLWST STACK FLAG: 0=F, +=E, -=B
1647 23600 067617 JMP WRITF (CODE 5 IS NOT USED)
1648 23601 077713 SBP *-2,S SET B FLAG
1649 23602 067612 JMP WRITX X;
1650* END OF FORMAT STATEMENT
1651 23603 162114 JSM FSPR,I /; UPDATE FORMAT POINTER
1652 23604 063351 WRITS JSM .CRLF
1653 23605 067573 JMP WRIT2
1654 23606 075713 WRITN SBP WRITS ANY NUMERIC SPEC FOUND?
1655 23607 074742 SBR 16
1656 23610 035455 STB FORMT CLEAR FORMAT NESTING FLAG
1657 23611 067723 JMP EWRI1 NO, GOTO STANDARD FORMAT
1658* X FIELD

581

568

1,444,141

568

PAGE 471

FORMATTED INPUT BLOCK EXECUTION

| | | | | | | |
|-------|--------------------------|--------|-------|-----|---------|------------------------------|
| 1659 | 23612 | 162114 | WRITX | JSM | FSPR,I | UPDATE FORMAT POINTER |
| 1660 | 23613 | 020012 | | LDA | .1 | |
| 1661 | 23614 | 027571 | | LDB | BLNKA | |
| 1662* | QUOTE FIELD | | | | | |
| 1663 | 23615 | 063374 | WRITQ | JSM | .OUTP | OUTPUT TO DEVICE |
| 1664 | 23616 | 067573 | | JMP | WRIT2 | |
| 1665* | | | | | | |
| 1666* | OUTPUT F,E, OR B FIELD | | | | | |
| 1667 | 23617 | 025466 | WRITF | LDB | LOCL1 | |
| 1668 | 23620 | 060563 | | JSM | SLWST | STACK FORMAT STMT POINTER |
| 1669 | 23621 | 162112 | WRIT4 | JSM | CKEOL,I | END OF WRITE STMT? |
| 1670 | 23622 | 074742 | | SBR | 16 | YES, CRLF NOT SUPRESSED |
| 1671 | 23623 | 067715 | | JMP | WRIT8-2 | YES |
| 1672 | 23624 | 162107 | | JSM | QTCHK,I | QUOTE NEXT IN WRITE |
| 1673 | 23625 | 067630 | | JMP | WRIT5 | NO |
| 1674 | 23626 | 063374 | | JSM | .OUTP | YES, OUTPUT IT |
| 1675 | 23627 | 067621 | | JMP | WRIT4 | |
| 1676* | | | | | | |
| 1677 | 23630 | 162110 | WRIT5 | JSM | PFORM,I | SAVE FLAGS AND |
| 1678 | 23631 | 061220 | | JSM | FETCH+1 | EVALUATE FORMULA |
| 1679 | 23632 | 060567 | | JSM | UNSTK | RECALL FORMAT STMT POINTER |
| 1680 | 23633 | 031466 | | STA | LOCL1 | |
| 1681 | 23634 | 060567 | | JSM | UNSTK | RECALL E,F, OR B FLAG |
| 1682 | 23635 | 031705 | | STA | FLOAT | |
| 1683 | 23636 | 070252 | | SAM | WRITB | SKIP IF B |
| 1684 | 23637 | 045466 | | ISZ | LOCL1 | |
| 1685 | 23640 | 121466 | | LDA | LOCL1,I | GET FIELD WIDTH SPEC |
| 1686 | 23641 | 045466 | | ISZ | LOCL1 | |
| 1687 | 23642 | 045466 | | ISZ | LOCL1 | |
| 1688 | 23643 | 025456 | WRITB | LDB | FORMS | |
| 1689 | 23644 | 074073 | | SBP | *+1,C | |
| 1690 | 23645 | 035456 | | STB | FORMS | SET NUMERIC SPEC FOUND FLAG |
| 1691 | 23646 | 125466 | | LDB | LOCL1,I | GET DECIMAL SPEC |
| 1692 | 23647 | 162114 | | JSM | FSPR,I | UPDATE FORMAT POINTERS |
| 1693 | 23650 | 031704 | | STA | WIDTH | |
| 1694 | 23651 | 070353 | | SAP | WRIT7 | SKIP IF E OR F |
| 1695* | OUTPUT B FIELD CHARACTER | | | | | |
| 1696 | 23652 | 160225 | | JSM | FLTRA,I | CONVERT TO ROUNDED INTEGER |
| 1697 | 23653 | 074742 | | SBR | 16 | OVERFLOW: USE 0 |
| 1698 | 23654 | 074117 | | LDA | B | |
| 1699 | 23655 | 052053 | | AND | BIT12 | MASK THE LOWER 12 BITS |
| 1700 | 23656 | 061353 | | JSM | TTY | OUTPUT SINGLE CHAR |
| 1701 | 23657 | 067711 | | JMP | WRIT6 | |
| 1702* | OUTPUT E OR F NUMBER | | | | | |
| 1703 | 23660 | 035703 | WRIT7 | STB | PLACE | |
| 1704 | 23661 | 021412 | | LDA | LSTPT | START TEMP BUFFER |
| 1705 | 23662 | 000061 | | ADA | .10 | |
| 1706 | 23663 | 070070 | | SIA | *+1 | |
| 1707 | 23664 | 070744 | | SAL | 1 | |
| 1708 | 23665 | 031400 | | STA | BADDR | ABOVE LOW STACK |
| 1709 | 23666 | 031706 | | STA | SBPTR | |
| 1710 | 23667 | 021413 | | LDA | HSTPT | |
| 1711 | 23670 | 070744 | | SAL | 1 | |
| 1712 | 23671 | 031623 | | STA | PEND | SET END OF BUFFER AT H-STACK |
| 1713 | 23672 | 021416 | | LDA | MODE | |
| 1714 | 23673 | 031641 | | STA | ST+1 | |

PAGE 472

FORMATTED INPUT BLOCK EXECUTION

582

| | | | | | | |
|-------|-------------------|--------|-------|-----|---------|---|
| 1715 | 23674 | 070742 | | SAR | 16 | |
| 1716 | 23675 | 031416 | | STA | MODE | SET FOR PROGRAM FORMAT |
| 1717 | 23676 | 024215 | | LDB | TMPOP | |
| 1718 | 23677 | 060406 | | JSM | XFAR1+1 | |
| 1719 | 23700 | 160234 | | JSM | NMOTA,I | OUTPUT A NUMBER TO THE BUFFER |
| 1720 | 23701 | 021641 | | LDA | ST+1 | |
| 1721 | 23702 | 031416 | | STA | MODE | |
| 1722 | 23703 | 021400 | | LDA | BADDR | |
| 1723 | 23704 | 011623 | | CPA | PEND | OVERFLOW? |
| 1724 | 23705 | 064561 | | JMP | MER8 | |
| 1725 | 23706 | 063372 | | JSM | .OUTP-2 | COMPUTE LENGTH AND OUTPUT THE
BUFFER |
| 1726 | 23707 | 025625 | | LDB | FIRST | |
| 1727 | 23710 | 043500 | | STB | BADDR | RESET BADDR |
| 1728 | 23711 | 162112 | WRIT6 | JSM | CKEOL,I | END OF STMT? |
| 1729 | 23712 | 074742 | | SBR | 16 | YES, CRLF NOT SUPRESSED |
| 1730 | 23713 | 067717 | | JMP | WRIT8 | YES |
| 1731 | 23714 | 067573 | | JMP | WRIT2 | NO |
| 1732* | END OF WRITE STMT | | | | | |
| 1733 | 23715 | 055412 | | DSZ | LSTPT | |
| 1734 | 23716 | 055412 | | DSZ | LSTPT | UNSTACK FLAGS |
| 1735 | 23717 | 070742 | WRIT8 | SAR | 16 | |
| 1736 | 23720 | 031455 | | STA | FORMAT | SET FORMAT COMPLETE FLAG |
| 1737 | 23721 | 067747 | | JMP | PR2+1 | |
| 1738* | | | | | | |
| 1739 | 23722 | 063471 | | JSM | .CTBL | |
| 1740 | 23723 | 025416 | EWRI1 | LDB | MODE | |
| 1741 | 23724 | 076150 | | RZB | *+3 | |
| 1742 | 23725 | 021622 | | LDA | TERM | |
| 1743 | 23726 | 072150 | | RZA | *+3 | |
| 1744 | 23727 | 160170 | | JSM | CLERA,I | |
| 1745 | 23730 | 162112 | | JSM | CKEOL,I | |
| 1746 | 23731 | 067745 | | JMP | PR2-1 | |
| 1747 | 23732 | 067745 | | JMP | PR2-1 | |
| 1748 | 23733 | 162104 | PR3 | JSM | PRINN,I | |
| 1749 | 23734 | 067746 | | JMP | PR2 | |
| 1750 | 23735 | 021625 | | LDA | FIRST | |
| 1751 | 23736 | 070076 | | TCA | | |
| 1752 | 23737 | 001400 | | ADA | BADDR | |
| 1753 | 23740 | 025625 | | LDB | FIRST | |
| 1754 | 23741 | 063374 | | JSM | .OUTP | |
| 1755 | 23742 | 021400 | | LDA | BADDR | |
| 1756 | 23743 | 031625 | | STA | FIRST | |
| 1757 | 23744 | 067733 | | JMP | PR3 | |
| 1758* | | | | | | |
| 1759 | 23745 | 074742 | | SBR | 16 | |
| 1760 | 23746 | 035622 | PR2 | STB | TERM | |
| 1761 | 23747 | 076150 | | RZB | PRI7 | |
| 1762 | 23750 | 162111 | | JSM | PRCHK,I | |
| 1763 | 23751 | 063351 | | JSM | .CRLF | |
| 1764 | 23752 | 164172 | PRI7 | JMP | XEC4A,I | |
| 1765* | | | | | | |
| 1766* | | | | | | |
| 1767* | | | | | | |
| 1768 | 23767 | | | ORG | 23767B | |
| 1769 | 23767 | 177563 | | ABS | EOUT-* | |
| 1770 | 23770 | 176564 | | ABS | EGET-* | GET EXECUTION |



570

1,444,141

570

PAGE 473

FORMATTED INPUT BLOCK EXECUTION

| | | | | | |
|-------|-------|--------|------|---------|----------------------------|
| 1771 | 23771 | 176314 | ABS | CMD—* | ASCII BUSS EXECUTION |
| 1772 | 23772 | 000000 | BSS | 1 | ** CHECKSUM WORD***** |
| 1773* | | | | | |
| 1774 | 23773 | 176022 | ABS | NFOPT—* | NON-FORMULA OPERATOR TABLE |
| 1775 | 23774 | 176061 | ABS | FUNNT—* | FUNCTION-NAME TABLE |
| 1776 | 23775 | 176017 | ABS | NULL—* | COMMAND—NAME TABLE (NULL) |
| 1777 | 23776 | 176006 | ABS | SYSNT—* | STATEMENT-NAME TABLE |
| 1778 | 23777 | 026000 | PAGE | OCT | 26000 |
| 1779* | | | | | |
| 1780 | | | END | | |

* NO ERRORS*

584

571

1,444,141

571

PAGE 474

CROSS-REFERENCE TABLE

| | | | | | | |
|-------|------|------|------|------|------|------|
| .1 | 0020 | 1108 | 1169 | 1466 | 1608 | 1660 |
| .10 | 0071 | 0072 | 1255 | 1460 | 1460 | 1705 |
| .100 | 0070 | | | | | |
| .1000 | 0069 | | | | | |
| .11 | 0035 | 0043 | | | | |
| .12 | 0036 | 1319 | | | | |
| .13 | 0039 | | | | | |
| .15 | 0048 | 0083 | 1630 | | | |
| .16 | 0049 | 0084 | 0671 | | | |
| .17 | 0610 | 0684 | | | | |
| .18 | 0555 | 0690 | | | | |
| .1E4 | 0068 | | | | | |
| .2 | 0027 | 1116 | 1212 | 1508 | | |
| .21 | 0050 | | | | | |
| .22 | 0051 | 0087 | | | | |
| .23 | 0052 | 0088 | | | | |
| .28 | 0053 | 1575 | | | | |
| .3 | 0028 | 0045 | 0889 | 1123 | 1153 | 1196 |
| .31 | 0054 | 0090 | 0904 | 1522 | | 1512 |
| .32 | 0055 | 1246 | 1640 | 1640 | | |
| .33 | 0056 | | | | | |
| .34 | 0057 | 0091 | | | | |
| .37 | 0058 | | | | | |
| .4 | 0029 | 1217 | | | | |
| .40 | 0059 | | | | | |
| .41 | 0060 | | | | | |
| .43 | 0061 | 0552 | | | | |
| .45 | 0062 | 0553 | | | | |
| .46 | 0063 | 0551 | | | | |
| .48 | 0064 | | | | | |
| .5 | 0030 | 1150 | | | | |
| .58 | 0065 | 0094 | | | | |
| .6 | 0031 | 0044 | | | | |
| .63 | 0066 | | | | | |
| .7 | 0032 | | | | | |
| .8 | 0033 | 0932 | 1188 | 1576 | | |
| .80 | 0606 | 1259 | | | | |
| .9 | 0034 | | | | | |
| .99 | 0074 | | | | | |
| .BADR | 0550 | | | | | |
| .BUFA | 0180 | | | | | |
| .CHAR | 1539 | 0957 | 1430 | | | |
| .CRLF | 1463 | 1652 | 1763 | | | |
| .CTBL | 1566 | 1018 | 1638 | 1739 | | |
| .EGEI | 1030 | 1033 | | | | |
| .EOUT | 1637 | 1621 | | | | |
| .FADA | 0185 | | | | | |
| .FDVA | 0188 | | | | | |
| .FLOP | 0547 | 0596 | | | | |
| .FLP | 0534 | 1161 | 1167 | 1168 | 1177 | 1178 |
| .FMPA | 0187 | | | | | 1180 |
| .FSBA | 0186 | | | | | |
| .FSR | 0190 | | | | | |
| .LIN | 0987 | 0962 | 0975 | 0980 | | |
| .LINI | 0983 | 0965 | 0973 | 0976 | 0978 | |



PAGE 475

CROSS-REFERENCE TABLE

[illegible]

CROSS-REFERENCE TABLE

[illegible]

589

574

1,444,141

574

PAGE 477

CROSS-REFERENCE TABLE

| | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|
| E | 0080 | | | | | | | | |
| E1 | 0235 | 1250 | | | | | | | |
| E10 | 0244 | 0711 | 0718 | | | | | | |
| E15 | 0249 | 0670 | 0673 | 0745 | 0750 | 0887 | 1107 | 1607 | |
| E20 | 0254 | | | | | | | | |
| E25 | 0259 | 0676 | 0677 | 0680 | 0683 | 0686 | 0692 | | |
| E30 | 0264 | | | | | | | | |
| E35 | 0269 | | | | | | | | |
| E40 | 0274 | 1556 | | | | | | | |
| E45 | 0279 | | | | | | | | |
| E5 | 0239 | 0819 | 0858 | 0941 | 0989 | 1165 | 1176 | | |
| E50 | 0284 | | | | | | | | |
| E71 | 1054 | 1047 | 1050 | | | | | | |
| E73 | 1379 | 1364 | 1370 | 1383 | 1391 | 1404 | | | |
| E79 | 1579 | 1441 | 1573 | | | | | | |
| EAND | 0844 | | | | | | | | |
| ECALA | 0215 | | | | | | | | |
| EFM3 | 1095 | 1072 | 1092 | | | | | | |
| EFMT | 1074 | 1048 | 1049 | | | | | | |
| EGE1 | 1041 | 1025 | | | | | | | |
| EGE2 | 1044 | 1097 | | | | | | | |
| EGE3 | 1045 | 1060 | 1064 | | | | | | |
| EGET | 1006 | 1770 | | | | | | | |
| EIAND | 0843 | 0586 | | | | | | | |
| EIOR | 0838 | 0587 | | | | | | | |
| ELINA | 0169 | | | | | | | | |
| ENIS | 0480 | | | | | | | | |
| ENTEA | 0218 | | | | | | | | |
| ENFTA | 0189 | | | | | | | | |
| EOCI | 0924 | 0935 | | | | | | | |
| EOCT | 0920 | 0594 | | | | | | | |
| EOL | 0038 | 0702 | 1290 | | | | | | |
| EOLCK | 0483 | 0912 | 0915 | | | | | | |
| EOST | 0481 | 0790 | | | | | | | |
| EOUT | 1625 | 1769 | | | | | | | |
| EROT | 0829 | 0588 | | | | | | | |
| ERRA | 0177 | | | | | | | | |
| ESCM | 0603 | 1284 | | | | | | | |
| ESKP | 1058 | 1052 | | | | | | | |
| ESLH | 1062 | 1046 | 1053 | | | | | | |
| ESTO | 0846 | 0836 | 0841 | | | | | | |
| EWRII | 1740 | 1620 | 1637 | 1657 | | | | | |
| EX | 0528 | 1405 | 1409 | 1417 | | | | | |
| EXP | 0401 | 0402 | | | | | | | |
| EXPOP | 0118 | | | | | | | | |
| F | 0081 | | | | | | | | |
| FDAAA | 0222 | | | | | | | | |
| FETCH | 0488 | 0852 | 0856 | 1163 | 1174 | 1554 | 1678 | | |
| FILLA | 0201 | | | | | | | | |
| FIRST | 0367 | 1726 | 1750 | 1753 | 1756 | | | | |
| FIX | 0505 | 1186 | 1208 | | | | | | |
| FLGBT | 0073 | 0854 | | | | | | | |
| FLOAT | 0386 | 1682 | | | | | | | |
| FLOOP | 0537 | 1021 | 1149 | 1151 | 1158 | 1184 | 1195 | 1201 | 1206 |
| | 1223 | 1225 | | | | | | 1211 | 1216 |
| FLOP | 0596 | 1020 | 1498 | 1612 | | | | | |

PAGE 478

CROSS-REFERENCE TABLE

| | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|
| FLTFA | 0193 | 0988 | | | | | | |
| FLTRA | 0192 | 0853 | 0857 | 1164 | 1175 | 1193 | 1555 | 1696 |
| FLTSA | 0195 | | | | | | | |
| FMTOP | 0609 | 1011 | 1616 | 1635 | | | | |
| FN | 0148 | | | | | | | |
| FND | 0462 | | | | | | | |
| FNDIA | 0224 | | | | | | | |
| FNDAA | 0223 | | | | | | | |
| FNDPS | 0476 | | | | | | | |
| FOPBS | 0206 | | | | | | | |
| FORMA | 0172 | 1104 | 1155 | | | | | |
| FORME | 0336 | | | | | | | |
| FORMS | 0335 | 1041 | 1043 | 1688 | 1690 | | | |
| FORMT | 0334 | 1535 | 1656 | 1736 | | | | |
| FPAG1 | 0766 | 0763 | | | | | | |
| FPAG2 | 0771 | 0761 | 0768 | | | | | |
| FPAGE | 0451 | | | | | | | |
| FROMA | 0122 | 0759 | | | | | | |
| FSCA | 0173 | 0681 | 0687 | 0693 | 0737 | 0871 | 0873 | |
| FSCE2 | 0441 | | | | | | | |
| FSCE4 | 0446 | | | | | | | |
| FSPR | 0647 | 1058 | 1062 | 1066 | 1085 | 1651 | 1659 | 1692 |
| FTMP1 | 0337 | | | | | | | |
| FTMP2 | 0338 | | | | | | | |
| FUNNT | 0615 | 1775 | | | | | | |
| FWAM | 0157 | | | | | | | |
| FWUP | 0296 | | | | | | | |
| FXFLA | 0194 | 0847 | 0939 | 0952 | 0960 | 1071 | 1205 | |
| FXIT | 1534 | 1134 | 1523 | | | | | |
| GETAD | 0461 | | | | | | | |
| GETCH | 0471 | 1232 | | | | | | |
| GETSK | 0475 | | | | | | | |
| GFLAG | 0404 | | | | | | | |
| GFRST | 0474 | | | | | | | |
| GNEXT | 0473 | 0667 | 0695 | 0784 | 0868 | | | |
| GPARM | 0424 | | | | | | | |
| GSIGN | 0486 | | | | | | | |
| GTCON | 0504 | | | | | | | |
| GTMP | 0472 | 1489 | | | | | | |
| GTOPP | 0466 | | | | | | | |
| GTSTA | 0213 | 0788 | | | | | | |
| HFLGI | 0321 | | | | | | | |
| HSTPT | 0302 | 1156 | 1263 | 1710 | | | | |
| I | 0067 | | | | | | | |
| ICNT | 0422 | | | | | | | |
| ID | 0565 | 0612 | 0612 | | | | | |
| IMPY | 0487 | 1598 | | | | | | |
| INF | 0077 | | | | | | | |
| INITS | 0456 | | | | | | | |
| INPT | 1429 | 1268 | 1277 | 1305 | 1444 | | | |
| INREA | 0226 | | | | | | | |
| INTCK | 0459 | | | | | | | |
| INTCV | 1606 | 1631 | | | | | | |
| INTEA | 0178 | | | | | | | |
| INTEL | 0340 | | | | | | | |
| INTFL | 0075 | | | | | | | |

588

576

CROSS-REFERENCE TABLE

| | | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|--|
| INTGR | 0402 | | | | | | | | | | | | |
| IOBUF | 0359 | | | | | | | | | | | | |
| IOHD | 0909 | 0663 | 0724 | 0732 | | | | | | | | | |
| IOHEA | 0229 | 0909 | | | | | | | | | | | |
| IOHED | 0649 | 0731 | | | | | | | | | | | |
| IOST | 1521 | 1456 | 1529 | 1542 | 1547 | | | | | | | | |
| IOSTA | 0225 | | | | | | | | | | | | |
| KEYFG | 0324 | | | | | | | | | | | | |
| L2T1 | 0347 | | | | | | | | | | | | |
| L2T2 | 0348 | | | | | | | | | | | | |
| LASTN | 0398 | | | | | | | | | | | | |
| LBOP | 0119 | | | | | | | | | | | | |
| LDEF | 0489 | | | | | | | | | | | | |
| LETCK | 0464 | | | | | | | | | | | | |
| LIMIT | 0465 | | | | | | | | | | | | |
| LIN | 0962 | 0592 | | | | | | | | | | | |
| LINEF | 1460 | 1465 | | | | | | | | | | | |
| LISTA | 0167 | | | | | | | | | | | | |
| LNFD | 0072 | 0968 | 1068 | 1233 | 1241 | 1278 | 1306 | | | | | | |
| LNINC | 0315 | | | | | | | | | | | | |
| LNUM | 0457 | | | | | | | | | | | | |
| LOCL1 | 0345 | 0925 | 0934 | 1074 | 1075 | 1076 | 1077 | 1080 | 1234 | 1267 | 1270 | | |
| | 1411 | 1413 | 1454 | 1457 | 1667 | 1680 | 1684 | 1685 | 1686 | 1687 | 1691 | | |
| LOCL2 | 0346 | | | | | | | | | | | | |
| LPCK | 0437 | 0665 | | | | | | | | | | | |
| LPCKE | 0438 | 0727 | 0870 | | | | | | | | | | |
| LPOP | 0120 | 1145 | | | | | | | | | | | |
| LROMA | 0123 | 0762 | | | | | | | | | | | |
| LSTAK | 0299 | | | | | | | | | | | | |
| LSTPT | 0301 | 1254 | 1499 | 1503 | 1610 | 1615 | 1704 | 1733 | 1734 | | | | |
| LTR | 0479 | 0886 | | | | | | | | | | | |
| LWAM | 0295 | | | | | | | | | | | | |
| M1 | 0124 | 0983 | 1115 | 1160 | 1293 | 1317 | 1323 | 1395 | 1437 | 1502 | 1509 | | |
| | 1582 | 1591 | | | | | | | | | | | |
| M10 | 0131 | 1266 | | | | | | | | | | | |
| M100 | 0144 | 1369 | | | | | | | | | | | |
| M1024 | 0146 | | | | | | | | | | | | |
| M11 | 0132 | | | | | | | | | | | | |
| M12 | 0133 | | | | | | | | | | | | |
| M13 | 0134 | | | | | | | | | | | | |
| M14 | 0135 | | | | | | | | | | | | |
| M15 | 0136 | | | | | | | | | | | | |
| M2 | 0125 | 1157 | 1611 | | | | | | | | | | |
| M23 | 0137 | | | | | | | | | | | | |
| M25 | 0138 | | | | | | | | | | | | |
| M256 | 0145 | 0924 | | | | | | | | | | | |
| M3 | 0126 | | | | | | | | | | | | |
| M32 | 0139 | | | | | | | | | | | | |
| M36 | 0140 | | | | | | | | | | | | |
| M4 | 0127 | 1614 | | | | | | | | | | | |
| M48 | 0141 | | | | | | | | | | | | |
| M5 | 0128 | 1224 | | | | | | | | | | | |
| M6 | 0129 | | | | | | | | | | | | |
| M64 | 0142 | | | | | | | | | | | | |
| M8 | 0598 | 1199 | | | | | | | | | | | |
| M80 | 0143 | | | | | | | | | | | | |

589

PAGE 480

CROSS-REFERENCE TABLE

| | | | | | | | |
|-------|------|------|------|------|------|------|------|
| M9 | 0130 | | | | | | |
| MASK1 | 0153 | | | | | | |
| MASK5 | 0154 | | | | | | |
| MAW | 0294 | 0766 | 0767 | | | | |
| MAXIN | 0400 | | | | | | |
| MAXSN | 0147 | | | | | | |
| MBIN1 | 0406 | | | | | | |
| MBIN2 | 0407 | | | | | | |
| MBOX1 | 0351 | | | | | | |
| MDIMA | 0208 | | | | | | |
| MEOV | 1250 | 1264 | | | | | |
| MER8 | 0445 | 1724 | | | | | |
| MF | 0524 | 0600 | 1334 | 1361 | 1388 | | |
| MINIT | 0455 | | | | | | |
| MODE | 0305 | 0849 | 1713 | 1716 | 1721 | 1740 | |
| MOUTA | 0638 | 0977 | | | | | |
| MSFLG | 0389 | | | | | | |
| MVTOH | 0477 | | | | | | |
| NAD1 | 1135 | 1181 | 1220 | | | | |
| NADD | 1132 | 1029 | 1034 | 1044 | 1226 | | |
| NBYB | 1239 | 1247 | 1322 | 1331 | 1394 | 1402 | 1406 |
| NBYT | 1232 | 1059 | 1067 | 1245 | | | |
| NDISP | 0382 | | | | | | |
| NDWRT | 0500 | | | | | | |
| NESC | 0708 | 0703 | | | | | |
| NEXT1 | 0450 | | | | | | |
| NEXTA | 0467 | | | | | | |
| NEXTL | 0449 | | | | | | |
| NFOP | 0896 | 0669 | 0682 | 0688 | | | |
| NFOPA | 0612 | 0896 | | | | | |
| NFOPT | 0578 | 0612 | 0612 | 1774 | | | |
| NMB1 | 1331 | 1339 | 1341 | 1355 | 1385 | | |
| NMB2 | 1343 | 1335 | 1337 | | | | |
| NMB3 | 1357 | 1333 | | | | | |
| NMB4 | 1366 | 1422 | | | | | |
| NMB5 | 1381 | 1358 | | | | | |
| NMB6 | 1387 | 1360 | | | | | |
| NMB7 | 1405 | 1415 | | | | | |
| NMB8 | 1417 | 1408 | | | | | |
| NMBR | 1314 | 1090 | | | | | |
| NMOTA | 0200 | 1719 | | | | | |
| NOEOF | 0482 | | | | | | |
| NPRIN | 0383 | | | | | | |
| NPWRT | 0499 | | | | | | |
| NULL | 0575 | 1776 | | | | | |
| NUM | 0396 | 0397 | 0657 | | | | |
| NUMCA | 0211 | | | | | | |
| NUMOA | 0212 | | | | | | |
| NXTDT | 0304 | | | | | | |
| NXTST | 0405 | | | | | | |
| OBLNK | 0468 | | | | | | |
| ODGIT | 0496 | | | | | | |
| OF | 0493 | | | | | | |
| ONE | 0045 | 1392 | | | | | |
| ONEXT | 0469 | 1286 | 1291 | | | | |
| OPCHI | 0646 | 1190 | | | | | |

578

1,444,141

578

PAGE 481

CROSS-REFERENCE TABLE

| | | | | | | | |
|-------|------|------|------|------|------|------|------|
| OPCHA | 0171 | | | | | | |
| OPDMK | 0079 | | | | | | |
| OPMSK | 0076 | 0658 | 1567 | | | | |
| OPTYP | 0397 | | | | | | |
| OTPT | 1454 | 0800 | 0805 | | | | |
| OTPTR | 0293 | | | | | | |
| OUTI | 1480 | 1470 | | | | | |
| OUTCR | 0470 | | | | | | |
| OUTIA | 0179 | | | | | | |
| OVCHA | 0216 | | | | | | |
| PAGE | 1778 | | | | | | |
| PARMA | 0306 | | | | | | |
| PARMN | 0307 | | | | | | |
| PBPTO | 0313 | | | | | | |
| PBPTR | 0310 | | | | | | |
| PBUFF | 0311 | | | | | | |
| PBUFO | 0312 | | | | | | |
| PEND | 0365 | 1113 | 1260 | 1289 | 1294 | 1712 | 1723 |
| PEXMA | 0165 | | | | | | |
| PFLAG | 0391 | | | | | | |
| PFOR | 1149 | 1137 | 1146 | | | | |
| PFORM | 0643 | 1677 | | | | | |
| PGIN0 | 0478 | | | | | | |
| PGINT | 0458 | | | | | | |
| PI/2 | 0044 | | | | | | |
| PLACE | 0384 | 1703 | | | | | |
| PLOTE | 0650 | 1014 | 1619 | | | | |
| PMODE | 0329 | | | | | | |
| PPFF | 0366 | | | | | | |
| PR17 | 1764 | 1761 | | | | | |
| PR2 | 1760 | 1737 | 1746 | 1747 | 1749 | | |
| PR3 | 1748 | 1757 | | | | | |
| PRADD | 0308 | 0807 | 0813 | 1133 | | | |
| PRCHK | 0644 | 0969 | 0997 | 1762 | | | |
| PRFLG | 0332 | | | | | | |
| PRINN | 0639 | 1748 | | | | | |
| PTRI | 0518 | 1111 | 1497 | | | | |
| QTCHA | 0230 | | | | | | |
| QTCHK | 0642 | 0801 | 0809 | 1672 | | | |
| QTOP | 0109 | 0161 | 0764 | | | | |
| QUOTE | 0561 | 0562 | | | | | |
| RBOP | 0117 | | | | | | |
| RBUFF | 0360 | | | | | | |
| RCOUT | 0333 | | | | | | |
| RDYDA | 0166 | | | | | | |
| RECO2 | 0749 | 0743 | | | | | |
| REED | 0161 | | | | | | |
| REEDL | 0339 | | | | | | |
| RES | 0417 | | | | | | |
| RESB1 | 0453 | | | | | | |
| RESBP | 0454 | | | | | | |
| RET1 | 1445 | 1435 | | | | | |
| RET2 | 0421 | | | | | | |
| RETN2 | 0420 | 0905 | | | | | |
| RETN3 | 0419 | 0877 | | | | | |
| RITE | 0637 | 0725 | 0733 | | | | |

CROSS-REFERENCE TABLE

[illegible]

CROSS-REFERENCE TABLE

| | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|------|--|
| STBLK | 0758 | 0661 | 0722 | 0781 | 0796 | 1006 | 1625 | | | | | |
| STEC | 0608 | 1171 | | | | | | | | | | |
| STEXP | 0443 | 0898 | | | | | | | | | | |
| STMAX | 0494 | | | | | | | | | | | |
| STORI | 0428 | | | | | | | | | | | |
| STORE | 0427 | 1209 | | | | | | | | | | |
| STPFL | 0323 | 1521 | | | | | | | | | | |
| STROP | 0430 | 0890 | | | | | | | | | | |
| STSRH | 0442 | | | | | | | | | | | |
| SUMPA | 0227 | | | | | | | | | | | |
| SVAR | 0817 | 0802 | 0810 | | | | | | | | | |
| SVARA | 0742 | 0701 | | | | | | | | | | |
| SYE12 | 0485 | | | | | | | | | | | |
| SYE25 | 0460 | | | | | | | | | | | |
| SYMC1 | 0433 | 0678 | | | | | | | | | | |
| SYMC2 | 0432 | | | | | | | | | | | |
| SYMCK | 0434 | | | | | | | | | | | |
| SYMTP | 0298 | | | | | | | | | | | |
| SYSNT | 0570 | 1777 | | | | | | | | | | |
| SYTSA | 0209 | | | | | | | | | | | |
| T1 | 0516 | 0831 | 0839 | 0844 | 0855 | 0971 | 0993 | 1010 | 1429 | | | |
| T2 | 0529 | 0656 | 0674 | 0708 | 0714 | 0716 | 0798 | 0840 | 0845 | 0859 | 1455 | |
| T3 | 0530 | | | | | | | | | | | |
| T5 | 0513 | 0728 | 0742 | 0749 | 0774 | 0782 | 0786 | 0817 | 1105 | 1117 | 1124 | |
| | 1486 | 1507 | 1511 | 1634 | | | | | | | | |
| T6 | 0531 | 1187 | 1198 | 1204 | 1210 | 1318 | 1340 | 1346 | 1365 | 1421 | | |
| T7 | 0532 | 1320 | 1348 | 1350 | | | | | | | | |
| T8 | 0514 | 0660 | 0897 | 1434 | 1469 | 1476 | 1589 | 1599 | | | | |
| T9 | 0533 | 1016 | 1024 | 1089 | 1091 | 1381 | 1387 | 1505 | 1506 | | | |
| TBEND | 0515 | 1473 | 1600 | | | | | | | | | |
| TD | 0523 | 1084 | 1093 | | | | | | | | | |
| TEMP | 0350 | 0351 | | | | | | | | | | |
| TEMP1 | 0352 | 0888 | | | | | | | | | | |
| TEMP2 | 0353 | 0891 | | | | | | | | | | |
| TEMP3 | 0354 | | | | | | | | | | | |
| TEMP4 | 0355 | | | | | | | | | | | |
| TEMPS | 0349 | 0806 | 0812 | 0860 | 0866 | 0875 | 1007 | 1132 | 1135 | 1142 | 1152 | |
| | 1154 | 1162 | 1170 | 1173 | 1179 | 1219 | 1222 | 1566 | 1570 | 1585 | 1626 | |
| | 1627 | 1632 | 1641 | | | | | | | | | |
| TERM | 0364 | 1000 | 1742 | 1760 | | | | | | | | |
| TFLAG | 0318 | | | | | | | | | | | |
| TFLGI | 0322 | | | | | | | | | | | |
| TFOR | 1184 | 1147 | | | | | | | | | | |
| TMPA | 0203 | | | | | | | | | | | |
| TMPOP | 0184 | 1717 | | | | | | | | | | |
| TMPRE | 0521 | 1114 | 1495 | 1504 | | | | | | | | |
| TNULL | 0527 | 1119 | | | | | | | | | | |
| TREGE | 0372 | | | | | | | | | | | |
| TSRHA | 0168 | | | | | | | | | | | |
| TSTPT | 0300 | | | | | | | | | | | |
| TTY | 0503 | 0970 | 0982 | 0998 | 1458 | 1480 | 1544 | 1548 | 1700 | | | |
| TW | 0520 | 1079 | 1086 | | | | | | | | | |
| TX | 0522 | 1087 | 1239 | 1244 | | | | | | | | |
| TYPEC | 0641 | 1045 | 1642 | | | | | | | | | |
| TYPFL | 0078 | | | | | | | | | | | |
| UFLAG | 0392 | | | | | | | | | | | |

PAGE 484

CROSS-REFERENCE TABLE

| | | | | | | | |
|-------|------|------|------|------|------|------|------|
| UKPTR | 0309 | | | | | | |
| UNITS | 0319 | | | | | | |
| UNNM | 1554 | 0944 | 0955 | 1008 | | | |
| UNSTK | 0448 | 1679 | 1681 | | | | |
| UPARA | 0191 | | | | | | |
| VALUE | 0297 | | | | | | |
| VAR01 | 0648 | 0751 | | | | | |
| VAROA | 0214 | 0675 | 0744 | | | | |
| WBUFF | 0423 | 1483 | | | | | |
| WIDTH | 0385 | 1693 | | | | | |
| WRIT2 | 1642 | 1653 | 1664 | 1731 | | | |
| WRIT4 | 1669 | 1675 | | | | | |
| WRIT5 | 1677 | 1673 | | | | | |
| WRIT6 | 1728 | 1701 | | | | | |
| WRIT7 | 1703 | 1694 | | | | | |
| WRIT8 | 1735 | 1671 | 1730 | | | | |
| WRITB | 1688 | 1683 | | | | | |
| WRITE | 0159 | | | | | | |
| WRITF | 1667 | 1647 | | | | | |
| WRITN | 1654 | 1643 | | | | | |
| WRITQ | 1663 | 1644 | | | | | |
| WRITS | 1652 | 1654 | | | | | |
| WRITX | 1659 | 1649 | | | | | |
| XC | 0412 | 1484 | 1485 | 1491 | 1496 | | |
| XEC4A | 0164 | 0808 | 0814 | 1537 | 1764 | | |
| XEC6A | 0221 | | | | | | |
| XFAR1 | 0425 | 1718 | | | | | |
| XFAR2 | 0429 | 0920 | 0943 | 0954 | 0987 | 1192 | 1393 |
| XFTOS | 1495 | 1487 | | | | | |
| XSCRA | 0220 | | | | | | |
| YC | 0414 | | | | | | |
| ZAP | 0152 | | | | | | |
| ZONK | 0491 | 1095 | | | | | |

MICROPROCESSOR

All of the above-listed routines and subroutines of basic instructions are implemented by the basic computing system shown in Figures 3A—B. Central control of this system is achieved by microprocessor 120. As shown in the block diagram of Figure 27 and in the detailed schematic diagram of Figures 28A—D, the microprocessor comprises a bipolar ROM 300 including seven ROM chips organized into 256 words of 28 bits. Eight J—K flip-flops contain the ROM address; (i.e. a 4-bit primary address and a 4-bit secondary address). A single chip 16-bit data selector permits any one of 16 different qualifier lines to be tested with a 4-bit qualifier code. This 4-bit qualifier code ROM chip serves a dual function in that it provides a complementing code to the 4 primary address flip-flops as well as selecting the proper qualifier to be tested. If branching in any ROM state is desired, the microinstruction BRC must also be given, BRC occurring with a QN (qualifier not met) signal from the data selector will cause the least significant bit of the address code to be inhibited to the secondary address flip-flop, thus causing the address to "branch" according to the state of the qualifier.

An additional feature of this ROM organization is the IQN microinstruction (inhibit if qualifier not met). When the IQN is given and the qualifier selected by the qualifier code is not met, the signal CCO (clock code zero) goes low. This inhibits all shift clock pulses from the clock decoder which in effect prevents execution of microinstructions in the ROM state.

To minimize the ROM word length, two 3-to-8 line decoders are used to expand 3 R-code outputs and 3 X-code outputs into a total of 14 microinstructions. Also the SCO and SCI outputs from ROM #5 are decoded in the Memory. The ALU outputs AC0, AC1, and AC2 are treated as address inputs to the ALU ROM and therefore need no decoding.

594

The microprocessor is responsible for the following:

1. Issuing a four-bit clock code to the clock decoder during each ROM state.
2. Issuing microinstructions to the memory, including the read and write microinstructions.
- 5 3. Issuing microinstructions to the shift registers for gating serial data into or out of the proper registers. 5
4. Issuing a four-bit ALU code to the Arith Logic Unit to select the proper binary or BCD arithmetic function.
- 10 5. Performing logical decisions (branching) based on the states of 16 qualifier inputs to the microprocessor. 10
6. Issuing next address information to the ROM address flip-flops in the microprocessor.
7. Transferring control to the input/output controller via the I/O strobe for execution of input or output instructions.
- 15 The full set of 28 ROM outputs with their associated microinstructions, the list of 16 qualifiers and assigned codes, and the microprocessor mnemonics are contained in the following tables: 15

| MICRO - INSTRUCTION SCT TABLE | | | | |
|-------------------------------|--|---------------------------------------|--|---|
| | POSITIVE
TRUE | NEGATIVE TRUE
OUTPUTS | | |
| CONTROL
FIELD | ROM
OUTPUT | DECODED μ -INSTRUCTION
-OUTPUT | | FUNCTION |
| GENERAL | 1. IQN
2. BRC
3. JTT
4. TTM
5. XTR | | | INHIBIT SHIFT CLOCK IF QUALIFIER NOT MET.
BRANCH: INHIBITS S00 IF QUALIFIER NOT MET.
T BUS $\rightarrow$ T REG
T BUS $\rightarrow$ M REG
A/B REG $\rightarrow$ R BUS |
| S-CODE | 6. SC1
7. SC0 | DECODED IN MEMORY | | ZERO $\rightarrow$ S BUS
M REG $\rightarrow$ S BUS
T REG $\rightarrow$ S BUS
ONE $\rightarrow$ S BUS |
| R-CODE | 8. RC2
9. RC1
10. RC0 | DECODED IN MICROPROCESSOR | | ONE $\rightarrow$ R BUS
P REG $\rightarrow$ R BUS
T REG $\rightarrow$ E REG $\rightarrow$ R BUS
STORE CONTENTS T REG $\rightarrow$ MEMORY
T BUS $\rightarrow$ Q REG (BIT 6)
Q REG $\rightarrow$ R BUS
READ MEMORY $<M> \rightarrow$ T REG
ZERO $\rightarrow$ R BUS |

| MICRO - INSTRUCTION SCT TABLE | | | | |
|---|------------------|--------------------------------------|-----|-----|
| | POSITIVE
TRUE | NEGATIVE TRUE
OUTPUTS | | |
| CONTROL
FIELD | ROM
OUTPUT | DECODED μ -INSTRUCTION
OUTPUT | | |
| X-CODE | 11. XC2 | DECODED IN MICROPROCESSOR | | |
| | 12. XC1 | XC2 | XC1 | XC0 |
| | 13. XC0 | 0 | 0 | 0 |
| | | TTQ | QAB | 1 |
| | | BCD | 0 | 1 |
| | | TBE | 0 | 1 |
| | | CAB | 1 | 0 |
| ALU | 14. AC2 | DECODED IN ALU | | |
| | 15. AC1 | AC2 | AC1 | AC0 |
| | 16. AC0 | 0 | 0 | 0 |
| | | XOR | 0 | 0 |
| | | AND | 0 | 0 |
| | | IOR | 0 | 1 |
| | | ZTT | 0 | 1 |
| | | ZTT . CBC | 1 | 0 |
| | | IOR . CBC | 1 | 0 |
| | | IOR . SBC | 1 | 1 |
| | | ADD | 1 | 1 |
| | | | 1 | 1 |
| | | | 1 | 1 |
| | | | 1 | 1 |
| FUNCTION | | | | |
| T BUS $\rightarrow$ Q REG | | | | |
| Q REG (BIT 11) $\rightarrow$ AB FLIP-FLOP | | | | |
| BCD ARITHMETIC MODE OF ALU | | | | |
| T BUS $\rightarrow$ E REG $\rightarrow$ R BUS | | | | |
| COMPLEMENT THE AB FLIP-FLOP | | | | |
| T BUS $\rightarrow$ P REG | | | | |
| T BUS $\rightarrow$ A/B REG | | | | |
| NONE OF THE ABOVE | | | | |
| EXCLUSIVE OR R $\oplus$ S $\rightarrow$ T BUS | | | | |
| LOGICAL AND R . S $\rightarrow$ T BUS | | | | |
| INCLUSIVE OR R + S $\rightarrow$ T BUS | | | | |
| ZERO $\rightarrow$ T BUS | | | | |
| ZERO $\rightarrow$ T BUS, CLEAR BINARY CARRY | | | | |
| INCLUSIVE OR, CLEAR BINARY CARRY | | | | |
| INCLUSIVE OR, SET BINARY CARRY | | | | |
| BINARY ADD | | | | |

8977

| MICRO - INSTRUCTION SET TABLE (Continued) | | |
|---|------------|--|
| CONTROL FIELD | ROM OUTPUT | FUNCTION |
| CLOCK | 17. CC1 | [THIS 4-BIT CODE INITIALIZES A PRESETTABLE DOWN COUNTER TO GENERATE ANY NUMBER OF SHIFT CLOCKS FROM 1 THROUGH 16. SHIFT IS INHIBITED BY IQN IF QUALIFIER NOT MET.] |
| | 18. CC2 | |
| | 19. CC4 | |
| | 20. CC8 | |
| QUALIFIER | 21. QC3 | [THIS 4-BIT CODE PERFORMS TWO FUNCTIONS:
(1.) ADDRESSING THE DATA SELECTOR TO SELECT ONE OF SIXTEEN QUALIFIER INPUTS.
(2.) PROVIDES COMPLEMENT CODE TO <u>PRIMARY</u> FLIP-FLOPS.] |
| | 22. QC2 | |
| | 23. QC1 | |
| | 24. QC0 | |
| SECONDARY ADDRESS | 25. S03 | [THIS 4-BIT CODE PROVIDES COMPLEMENT CODE TO THE <u>SECONDARY</u> FLIP-FLOPS. IF BRC IS GIVEN AND QUALIFIER IS NOT MET, THE S00 BIT IS INHIBITED.] |
| | 26. S02 | |
| | 27. S01 | |
| | 28. S00 | |

SPECIAL MICRO - INSTRUCTIONS:

TQR = UTR . XTR TRANSFERS
Q-REGISTER TO PRIMARY ADDRESS AS SHOWN

Q14 . Q11 → PO4
Q12 → PO5
Q13 → PO6
Q14 → PO7

IOS = PTR . XTR INITIATES:

- a) TRANSFER IF CONTROL TO I/O IF Q10 = 1
b) SETS "SINGLE SERVICE" FF IN I/O VIA SRA IF Q10 = 0

SPECIAL OPERATIONS:

BCD SUM → A < 0-3 > = BCD . UTR . TOM CLOCK

CLEAR DECIMAL CARRY = QAB . ROM CLOCK

SET DECIMAL CARRY = UTR . BCD . ROM CLOCK

DECIMAL ADD = BCD . ZTT T < 0-3 > + A < 0-3 > → Q < 0-3 >

10's COMPLEMENT W/DECIMAL ADD = BCD . ADD T < 0-3 > + A < 0-3 > → Q < 0-3 >

QUALIFIER SET TABLE

QUALIFIER CODE

| | QC3 | QC2 | QC1 | QC0 | MNEMONIC | FUNCTION | |
|----|-----|------|-----|-----|----------|--|----|
| | | 0000 | | | Q00 | Shift/Skip One Bit | |
| 5 | | 0001 | | | Q01 | Shift/Skip Two Bits | 5 |
| | | 0010 | | | Q02 | Shift/Skip Four Bits | |
| | | 0011 | | | Q03 | Shift/Skip Eight Bits | |
| | | 0100 | | | Q04 | Fast Square Root Qualifier | |
| | | 0101 | | | Q05 | Set Bit in A/S Group; FDV qualifier | |
| 10 | | 0110 | | | Q06 | T-Bus Qualifier via TQ6 | 10 |
| | | 0111 | | | QBC | Binary Carry from ALU | |
| | | 0110 | | | QBC | Binary Carry from ALU | |
| | | 1000 | | | Q07 | P Register, Bit 0, for BCD counting | |
| | | 1001 | | | Q15 | Indirect Address, Clear Bit in P/S Group | |
| 15 | | 1010 | | | QMR | Memory Reference qualifier | 15 |
| | | 1011 | | | Q10 | Current Page Qualifier, FXA Qualifier | |
| | | 1100 | | | QNR | Non-Service Request Qualifier | |
| | | 1101 | | | Q08 | FMP Qualifier | |
| | | 1110 | | | QDC | Decimal Carry from ALU | |
| 20 | | 1111 | | | *QRD | ROM Disable (Normally Zero) | 20 |

POP will preset ROM address Flip-Flops at turn-on.

*QRD may be used with IQN to insure zero shift except when in I/O loop.

MICROPROCESSOR MNEMONICS

| | | | |
|----|--------------------------|--|----|
| 25 | <u>Clock Signals</u> | Memory Clock | 25 |
| | MCK | Shift Clock | |
| | SCK | External Clock | |
| | XTC | ROM Clock for Flip Flops | |
| | RCF | ROM Clock for Address Flip Flops | |
| 30 | RCA | Inhibit Internal Clock | 30 |
| | IIC | Inhibit Clock | |
| | INH | Inhibit ROM Clock (Also primary and secondary Flip Flop) | |
| | IPS | | |
| 35 | CC8 | Clock Code: Binary Code that programs the number of shift clocks | 35 |
| | CC4 | | |
| | CC2 | | |
| | CC1 | | |
| | CC0 | Inhibits Shift Clocks | |
| | <u>Address Mnemonics</u> | | |
| | POP | Power on Preset | 40 |
| | IQN | Inhibit if Qualifier not met | |
| | BRC | Branch | |

MICROPROCESSOR MNEMONICS (Continued)

Q-Register

| | | | |
|----|---|--|----|
| | TQR | Transfer Q11, Q12, Q13, Q14 to primary
addr. Flip Flops | |
| 5 | TTQ | T-Bus to Q-Register | 5 |
| | QTR | Q-Register to R-Bus | |
| 10 | Q10
Q9
Q8
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Q0 | Bits 10 — 0 of Q-Register | 10 |
| 15 | | | 15 |

Data Qualifiers

| | | | |
|----|-----|---|----|
| | QP0 | Bit 0 of P-Register | |
| 20 | QRD | Qualifier ROM Disable (I/O interrupt) | 20 |
| | QNR | Qualifier No Request (Keyboard Interrupt) | |
| | QDC | Decimal Carry | |
| | QBC | Binary Carry | |

Memory

| | | | |
|----|------------|---------------------|----|
| 25 | SC0
SCI | S-Bus Code | 25 |
| | TTT | T-Bus to T-Register | |
| | TTM | T-Bus to M-Register | |
| | RDM | Read Memory | |
| 30 | WTM | Write Memory | 30 |

A, B, P, E-Registers

| | | | |
|----|-----------|---|----|
| | QAB | Q-Register to AB Flip Flop | |
| | AB=0 | A-Register Operation | |
| | AB=0 | B-Register Operation | |
| 35 | TTM (ROM) | T-Bus to A or B Register
(Originates at ROM Decoder) | 35 |
| | XTR | A or B Register to R-Bus | |
| | TTP | T-Bus to P-Register | |

| | |
|---|---|
| $\overline{\text{PTR}}$ | P-Registers to $\overline{\text{R}}$ -Bus |
| $\overline{\text{TBE}}$ | T-Bus to E-Register to $\overline{\text{R}}$ -Bus |
| $\overline{\text{TRE}}$ | T-Register to E-Register |
| $\overline{\text{UTR}}$ | Units to $\overline{\text{R}}$ -Bus |
| $\left. \begin{array}{l} \text{AC2} \\ \text{AC1} \\ \text{AC0} \end{array} \right\}$ | Arithmetic Codes for Arithmetic Logic Unit |
| $\overline{\text{BCD}}$ | Decimal Arithmetic |
| $\overline{\text{SDR}}$ | Disables ROMs for Single Step Tester Operation |

End of the ROM chips of Figures 27 and 28A—D is organized into 256 words of 4 bits each constructed in accordance with the following table, where each "L" represents a low (or "0") state and each "H" represents a high or ("1") state:

[illegible]

ROM CHIP 5

[illegible]

603

[illegible]

592

[illegible]

605

[illegible]

ROM CHIP 9

| | | | | | | | | | |
|------|---------|------|------|------|------|------|------|------|------|
| 0238 | 000—007 | HHLH | HHLL | HHLL | LHLH | HHLL | HHHH | HHLH | HHLH |
| 0239 | 008—015 | HHLL | HHHH | LHLH | HHLL | HLHH | HHLL | HHLL | LHLL |
| 0240 | 016—023 | HHLL | HHLL | HHLL | HHHH | HHLL | HHLL | HHLL | HHLL |
| 0241 | 024—031 | HHLL | HHHH | HHHL | LHLL | HLHH | LHLL | HHHL | HHLL |
| 0242 | 032—039 | HLLH | HHLL | LHHH | HHLL | LHHH | LHLL | HHLL | HHLL |
| 0243 | 040—047 | HHLL | HHHH | HHLL | HHHH | HHHH | HHLL | HHLL | HHLL |
| 0244 | 048—055 | LHLL | HHLL | HHLL | HHLH | HHLL | HHLL | LHLL | LHHH |
| 0245 | 056—063 | HHLL | HHHH | HHLL | HHLH | HHLL | HHLL | HHHH | LHLL |
| 0246 | 064—071 | HHLH | HHLH | HHLL | HHLL | HHLL | LHLL | HHHH | LHLL |
| 0247 | 072—079 | HLLL | HLHH | HHLL | HHLL | HHHL | LHLL | HHLL | LHLL |
| 0248 | 080—087 | HHLL | HHLL | HHLL | HHLL | LHLL | HHHH | HHHL | HHLL |
| 0249 | 088—095 | HLHH | HLHH | HHLL | HHLL | HHLL | HHHH | HHHL | LHLL |
| 0250 | 096—103 | HLLL | HHLL | HHLL | HHLL | HHLL | HHLL | HHLL | HHLL |
| 0251 | 104—111 | HHHH | HLHH | HHLH | LHLL | HHLL | LHLL | HHHL | HHLL |
| 0252 | 112—119 | HLLL | HHLL | HHLL | HHLL | HHLL | HHLL | HHLL | HHLL |
| 0253 | 120—127 | HHLH | HHHH | HHHH | HHLL | HHLH | HHHL | LHLL | LHLL |
| 0254 | 128—135 | HHLH | LHHH | HHLL | HHLL | HHLL | HHLL | LHLL | LHLL |
| 0255 | 136—143 | HHHH | HHHL | LHLL | HHLL | HHLL | HHLL | LHLL | HHLL |
| 0256 | 144—151 | HLLH | HHHH | HHLL | HLHH | HHLL | HLLH | HHHH | HHLL |
| 0257 | 152—159 | HHLL | HHLL | HHLL | HHLL | HLLH | HHLL | HHLL | LHLL |
| 0258 | 160—167 | HHLH | HHHL | LHLL | LHHH | LHLL | HHHH | LHLL | HHLL |
| 0259 | 168—175 | HHHH | LHLL | HHHH | HHLL | LHLH | LHLL | LHLL | HHLL |
| 0260 | 176—183 | HLLH | LHLL | HHLH | HHHL | LHHH | LHHH | HHLL | HHLL |
| 0261 | 184—191 | HHLL | HHLL | LHLL | HHHL | LHHH | HHLL | HHLL | LHLL |
| 0262 | 192—199 | HHHL | HHLL | LHLH | LHLL | LHLL | HHHL | LHLL | LHLL |
| 0263 | 200—207 | HHHH | HHLL | HLHL | LHHH | LHLL | LHLL | HHLL | LHLL |
| 0264 | 208—215 | HHHL | HLHH | HHLL | HLHH | HLLH | HHHH | HHHL | LHLL |
| 0265 | 216—223 | HHLL | HHLL | HHLL | HHHH | LHHH | LHLL | LHLL | HHLL |
| 0266 | 224—231 | LHHH | HHHH | LHLL | LHLL | HHLL | HHLL | LHLL | HHLL |
| 0267 | 232—239 | HHHH | HHLL | HHLL | LLHH | HHLL | HHLL | HHLL | LHLL |
| 0268 | 240—247 | HHHH | HHLL | LHHH | HHLL | LHHH | HHHH | LHHH | LHLL |
| 0269 | 248—255 | HLLL | HHLL | HHLL | HHHH | HLLL | HHLL | HHLL | HHLL |
| 0270 | | | | | | | | | |

Each of the 71 basic instructions employed by the calculator is implemented by one or more of the above-described microinstructions and associated control signals issued by the microprocessor. The manner in which this is accomplished is shown and described in detail in the flow charts of Figures 29A—H. Each rectangular box of these flow charts represents a state of ROM 300 of the microprocessor and includes the mnemonic of the microinstructions and control signals stored in that ROM state. The number at the upper right-hand corner of each of these rectangular boxes represents the number of shift clock pulses required by the microinstructions of that ROM state. A simplified overview of these detailed flow charts is shown in Figures 6A—B.

PROGRAMMABLE CLOCK

Given a computing system organized to process binary data serially and under control of microinstructions stored in ROM 300 as shown in Figures 3A—B and 27, the implementation of a general purpose instruction set requires that some number of bits be shifted into or out of the storage registers. Depending on the operation being performed, the number of bits may vary from zero to n , where n is the number of bits in a single machine word.

If each clock period of the ROM clock corresponds to a one bit shift, a count loop must be employed to provide the desired number of shifts. A rather large number of such count loops would exist in order to implement an entire instruction set. An alternative method is to provide additional hardware which permits assignment of the desired number of shifts in a single state of ROM 300. Such an arrangement requires a variable cycle time for each state of ROM 300, but results in a very substantive saving in total number of ROM states.

To implement a variable number of shift clocks in a single state of the microprocessor, two separate clocks are required. The shift clock is applied to the data

607

storage registers in the memory, the shift register block, the arithmetic logic unit and the input/output block. The ROM clock is applied to the ROM address flip-flops in the microprocessor, and occurs once for each state in the microprogram. The number of shift clock pulses that occurs in any given ROM state is determined by a 4-bit clock code sent to the clock decoder from the microprocessor.

If no shift clocks are desired, a separate signal $\overline{CC0}$ from the microprocessor inhibits the shift clock output, independent of the clock code issued in that state. In this way, any number of shifts between and including zero and 16 may be implemented with a 4-bit clock code and an inhibit signal.

This inhibit signal offers an additional powerful feature when gated by the qualifier test logic in the microprocessor as shown in Figure 3A. The qualifier test logic includes a 4-bit qualifier code from ROM 3 that selects one of 16 qualifier inputs to the data selector. The data selector output QN (qualifier not met) will be high if the selected qualifier input was low. By using the QN signal to gate the inhibit microinstruction, IQN, the shift clock will be inhibited only when the qualifier is not met. Thus, all microinstructions requiring shift clocks that are issued in a given ROM state may be either executed or inhibited, depending on the logical state of the qualifier under test.

The ROM clock is applied to the eight J—K flip-flops which address the 256 word microprocessor ROM. During any given state, the complementing (J—K) inputs to the 4 primary address flip-flops are set up by the qualifier code or q-register code. The 4 secondary address flip-flop inputs are determined by the ROM 4 outputs, the BRC microinstruction, and the data selector output QN. Where ROM clock goes low, the negative edge-triggered flip-flops will cause transition of the ROM address to the next ROM state.

As shown in the block diagram of Figure 30 and the detailed schematic diagram of Figures 31A—C, a crystal controlled system clock output is inverted to generate memory clock, MCK. This signal is again inverted to clock a D flip-flop having an output (control clock), which will go low if the end-of-count signal (borrow) from the down counter has occurred at the D input. The ROM clock will also go low at this time, initiating a new ROM state in the microprocessor. Control clock will normally remain low for one system clock period, and in turn generates a load signal which is delayed a half period from control clock by means of a second D flip-flop. The 4-bit clock code from the microprocessor is preset into the counter while the load signal is low.

As the load signal goes high, ROM clock also goes high, completing the fixed interval portion of ROM clock and shift clock as shown in Figure 32. A series of clock pulses are now gated onto shift clock, SCK, until the preset counter has counted down to zero, causing control clock to again go low, completing the ROM cycle.

The inhibit signal, INH, from memory may lengthen the normal fixed interval of ROM clock by clearing the D flip-flop and holding control clock low. This may occur during memory refresh or external test operations. In this situation, the counter remains preset and the correct number of shifts will be generated when the inhibit goes away.

SHIFT REGISTER UNIT

As shown in the detailed schematic diagrams of Figures 28A—D and 33A—D, A-register 122, B-register 124, P-register 126, Q-register 128, and E-register 130 of Figures 3A—B comprise bipolar status registers, the contents of which are recirculated when data is output to the R-bus or the S-bus. Full control of these registers in use and type of operations performed is maintained by the microinstructions from the microprocessor. The number of bits to be shifted in any one ROM state of the microprocessor is determined by the number of shift clocks from the clock decoder. This shift clock appears at the shift clock input of each shift register that is enabled by the microprocessor during that ROM cycle.

ARITHMETIC LOGIC UNIT

The development of complex read-only memory arrays on a single chip have made possible a hardware implementation of central processing units (CPUs) and arithmetic logic units (ALUs) with far fewer components than were previously possible. In this application, two bipolar read-only memory chips are combined with carry flip-flops and adapted to perform one-bit binary logic and arithmetic operations as well as four-bit binary-coded-decimal (BCD) arithmetic operations.

The two bipolar read-only memory chips may comprise, for example, Hewlett-Packard 16-pin dual-in-line packaged bipolar ROMs organized into 256 words by 4-bits and of the same type as shown and described in U.S. Patent No. 3,721,964.

The binary/BCD Arithmetic Logic Unit consists of five integrated circuits connected as shown in the block diagram of Figure 34 and the detailed schematic diagram of Figures 33A—D. Specifically, the packages consist of two 1024-bit ROMs, a dual D-type flip-flop and two quad two-input NAND gates.

Internally the desired binary logical function, binary arithmetic operation or BCD operation is selected by the ALU code as shown below.

| ALU CODE | | | | ALU FUNCTION | DESCRIPTION |
|------------------|-----|-----|-----|--------------|--|
| BCD | AC2 | AC1 | AC0 | | |
| Binary Functions | 0 | 0 | 0 | XOR | Exclusive OR $\dots R \oplus S \rightarrow T$ |
| | 0 | 0 | 1 | AND | Logical AND $\dots R \cdot S \rightarrow T$ |
| | 0 | 0 | 1 | IOR | Inclusive OR $\dots R + S \rightarrow T$ |
| | 0 | 0 | 1 | ZTT | Zero $\rightarrow$ T-BUS |
| | 0 | 1 | 0 | ZTT, CBC | Zero $\rightarrow$ T-BUS, Clear Binary Carry |
| | 0 | 1 | 0 | IOR, CBC | Inclusive OR, Clear Binary Carry |
| | 0 | 1 | 1 | IOR, SBC | Inclusive OR, Set Binary Carry |
| | 0 | 1 | 1 | ADD | Binary ADD $\dots B + S + BC \rightarrow T, C$ |
| BCD Functions | 1 | 0 | 1 | BCD ADD | BCD ADD $\dots T_{0-3} + A_{0-3} \rightarrow \Sigma_{0-3}$ |
| | 1 | 1 | 1 | BCD COMP/ADD | IO's Complement and BCD ADD |

ALU FUNCTION CODE ASSIGNMENTS

The function code input "BCD" selects between the binary mode and BCD mode of operation.

In the binary mode, the function code inputs AC0, AC1, and AC2 select the desired logical function or arithmetic operation. The binary input data enters ROM #1 on the carry, S-bus and R-bus input lines, and the binary result appears on the T-bus and binary carry output lines. ROM #2 is not used in the binary mode.

In the BCD mode of operation, the two function code lines AC0 and AC1 are disabled from the Micro-processor and these two lines carry the T02 and T03 bits of BCD data from the T-Register. The ALU function code line AC2 is used to select the desired BCD operation. If AC2 is low, the four-bit output $\Sigma 0, \Sigma 1, \Sigma 2, \Sigma 3$

609

will be the BCD sum of the two BCD data inputs. If AC2 is high and decimal carry has been set, the four-bit output $\Sigma 0, \Sigma 1, \Sigma 2, \Sigma 3$ will be the BCD Tens Complement of the BCD data from the T-Register. In the BCD mode, the binary carry output will be disabled and the decimal carry output will be enabled to ROM #1.

Although only one-fourth of the available registers in ROM #1 are required for the eight binary operations, the concept of adding a second 1024-bit ROM to perform the BCD operations grew from several basic concepts:

- 1) The least significant BCD sum bit, $\Sigma 0$, is always identical to the binary sum bit; therefore, only three additional outputs, $\Sigma 1, \Sigma 2$, and $\Sigma 3$ need be generated. For BCD complement operations, the decimal carry flip-flop defines whether or not the least significant bit should be complemented.
- 2) In forming the "nine's complement" of the T-Register BCD data in ROM #1, it can be seen that for 8421 code the second least significant bit T01 is the same before and after forming the complement. Thus only two bits, T02 and T03 need be complemented prior to input into ROM #2. The ten's complement with add is then found by presetting decimal carry and performing a BCD sum of the three most significant digits in ROM #2.
- 3) With only eight ROM inputs available, some sharing of inputs is required for ROM #1. During binary operations, all four function codes and only one bit of T-Register data is required. During BCD operations, all four bits of T-Register data and only two function codes are required. Use of two NAND gates in wire-OR connection with the open collector function codes AC0 and AC1 permits sharing of the two inputs.

This arrangement left one input still available to ROM #2. By programming this input to always make output DC1 true, the micro-instruction UTR can serve two purposes — placing units on the R-bus and also set decimal carry if BCD is true. When BCD is false, clock is inhibited to decimal carry. This feature permits saving decimal carry information during all binary operations. Similarly, binary carry is saved during the four binary operations AND, IOR, XOR, and ZTT by connecting AC2 such that when AC2 is false the shift clock is inhibited to the binary carry flip-flop.

In summary, the mode select input "BCD" performs the following functions:

- 1) Addresses the proper 128 word set of word lines in ROM #1.
- 2) Enables the T02 and T03 data lines to ROM #1 only in BCD mode.
- 3) Enables clock to decimal carry flip-flop only in BCD mode.
- 4) Selects binary carry or decimal carry into ROM #1 as appropriate.
- 5) Transfers outputs $\Sigma 0, \Sigma 1, \Sigma 2, \Sigma 3$, to A-Register only in BCD mode.

The remaining three ALU function codes select the proper set of word lines in ROM #1 to perform the eight binary functions. In addition, the AC2 input performs the following functions.

- 1) Enables clock to binary carry flip-flop only during the four carry-related binary functions and the BCD comp/add function.
- 2) In the BCD mode, AC2 causes BCD data bit T00, T02 and T03 to convert to nine's complement form.

The ALU has a total of 15 inputs which include 8 data inputs, 2 clock inputs and 5 microinstructions. Four data output lines are required, and two additional output lines from carry flip-flops are available as qualifier inputs to the microprocessor. The ALU and shift register mnemonics are listed in the following table:

SHIFT REGISTERS & ALU BOARD MNEMONICS

| | |
|------------|---|
| TRE | T-Register to E-Register to R-Bus |
| TO0 | Bit 0 of T-Register |
| TBE | T-Bus to E-Register to R-Bus |
| TTX — TEST | T-Bus to A/B-Register from Tester |
| TTX — I/O | T-Bus to A/B-Register from I/O
(Board #12) |
| TTX — ROM | T-Bus to A/B-Register from Processor
(Board #13) |

6-10

598

1,444,141

598

SHIFT REGISTERS & ALU BOARD MNEMONICS (Continued)

| | | | |
|----|---------------------------|--|----|
| | $\overline{\text{TTX}}$ | Logical "OR" of Three $\overline{\text{TTX}}$ Signals | |
| | AB | Status of AB-Flip-Flop
AB = 0 A-Reg. Operation
AB = 1 B-Reg. Operation | 5 |
| 5 | $\overline{\text{XTR}}$ | A/B Register to $\overline{\text{R}}$ -Bus | |
| | $\overline{\text{UTR}}$ | Logical "I" to $\overline{\text{R}}$ -Bus | |
| | $\overline{\text{TQR}}$ | Q-Register to Primary Address Flip-Flop | |
| | $\overline{\text{AB}}$ | Complement of AB | |
| 10 | $\overline{\text{TTP}}$ | T-Bus to P-Register | 10 |
| | SCK | Shift Clock | |
| | QP0 | Qualifier, Bit 0 of P-Register | |
| | $\overline{\text{PTR}}$ | P-Register to $\overline{\text{R}}$ -Bus | |
| | QO0 | Q-Register Bit 0 | |
| 15 | $\overline{\text{QTR}}$ | Q-Register to $\overline{\text{R}}$ -Bus | 15 |
| | RCK | ROM Clock | |
| | QAB | Q-Register to AB-Flip-Flop, also clears decimal carry. | |
| | SCB | Set Binary Carry | |
| | $\overline{\text{BCD}}$ | Decimal Arithmetic | |
| 20 | AC2 | ALU Operation Code | 20 |
| | QBC | Qualifier, Binary Carry | |
| | $\overline{\text{S-Bus}}$ | Data Bus | |
| | AC1 | ALU Operation Code | |
| | AC0 | ALU Operation Code | |
| 25 | TO2 | Bit 2 of T-Register | 25 |
| | TO3 | Bit 3 of T-Register | |
| | SDR | Signal to Disable ROMs | |
| | TO1 | Bit 1 of T-Register | |
| | T-Bus | Data Bus | |
| 30 | ALU | Arithmetic Logic Unit | 30 |

(—) Indicates Negative True Signal

The following table gives an example of how the two ALU ROM chips shown in Figures 33A—D and 34 can be constructed to implement the above described ALU functions (in this table each "1" represents a "low" state and each "0" represents a "high" state):

35

35

611

ROM #1

| | | | | | | | |
|------|-------|------|-------|------|-------|------|------|
| / | 1000; | 1/ | 0000; | 2/ | 0000; | 3/ | 1000 |
| 4/ | 1000; | 5/ | 0000; | 6/ | 0000; | 7/ | 1000 |
| 8/ | 0000; | 9/ | 0000; | 10/ | 0000; | 11/ | 1000 |
| 12/ | 0000; | 13/ | 0000; | 14/ | 0000; | 15/ | 1000 |
| 16/ | 0000; | 17/ | 1000; | 18/ | 1000; | 19/ | 1000 |
| 20/ | 0000; | 21/ | 1000; | 22/ | 1000; | 23/ | 1000 |
| 24/ | 1000; | 25/ | 1000; | 26/ | 1000; | 27/ | 1000 |
| 28/ | 1000; | 29/ | 1000; | 30/ | 1000; | 31/ | 1000 |
| 32/ | 1000; | 33/ | 0000; | 34/ | 0000; | 35/ | 1000 |
| 36/ | 1000; | 37/ | 0000; | 38/ | 0000; | 39/ | 1000 |
| 40/ | 0000; | 41/ | 0000; | 42/ | 0000; | 43/ | 1000 |
| 44/ | 0000; | 45/ | 0000; | 46/ | 0000; | 47/ | 1000 |
| 48/ | 0000; | 49/ | 1000; | 50/ | 1000; | 51/ | 1000 |
| 52/ | 0000; | 53/ | 1000; | 54/ | 1000; | 55/ | 1000 |
| 56/ | 1000; | 57/ | 1000; | 58/ | 1000; | 59/ | 1000 |
| 60/ | 1000; | 61/ | 1000; | 62/ | 1000; | 63/ | 1000 |
| 64/ | 1100; | 65/ | 1100; | 66/ | 1100; | 67/ | 1100 |
| 68/ | 1100; | 69/ | 1100; | 70/ | 1100; | 71/ | 1100 |
| 72/ | 0000; | 73/ | 0000; | 74/ | 0000; | 75/ | 1000 |
| 76/ | 0000; | 77/ | 0000; | 78/ | 0000; | 79/ | 1000 |
| 80/ | 0100; | 81/ | 0100; | 82/ | 0100; | 83/ | 1100 |
| 84/ | 0100; | 85/ | 0100; | 86/ | 0100; | 87/ | 1100 |
| 88/ | 0000; | 89/ | 1000; | 90/ | 1000; | 91/ | 0100 |
| 92/ | 1000; | 93/ | 0100; | 94/ | 0100; | 95/ | 1100 |
| 96/ | 1100; | 97/ | 1100; | 98/ | 1100; | 99/ | 1100 |
| 100/ | 1100; | 101/ | 1100; | 102/ | 1100; | 103/ | 1100 |
| 104/ | 0000; | 105/ | 0000; | 106/ | 0000; | 107/ | 1000 |
| 108/ | 0000; | 109/ | 0000; | 110/ | 0000; | 111/ | 1000 |
| 112/ | 0100; | 113/ | 0100; | 114/ | 0100; | 115/ | 1100 |
| 116/ | 0100; | 117/ | 0100; | 118/ | 0100; | 119/ | 1100 |
| 120/ | 0000; | 121/ | 1000; | 122/ | 1000; | 123/ | 0100 |
| 124/ | 1000; | 125/ | 0100; | 126/ | 0100; | 127/ | 1100 |
| 128/ | 0000; | 129/ | 0000; | 130/ | 0000; | 131/ | 0000 |
| 132/ | 0000; | 133/ | 0000; | 134/ | 0000; | 135/ | 0000 |
| 136/ | 0010; | 137/ | 1010; | 138/ | 1010; | 139/ | 0110 |
| 140/ | 1010; | 141/ | 0110; | 142/ | 0110; | 143/ | 1110 |
| 144/ | 0001; | 145/ | 1001; | 146/ | 1001; | 147/ | 0101 |
| 148/ | 1001; | 149/ | 0101; | 150/ | 0101; | 151/ | 1101 |
| 152/ | 0011; | 153/ | 1011; | 154/ | 1011; | 155/ | 0111 |
| 156/ | 1011; | 157/ | 0111; | 158/ | 0111; | 159/ | 1111 |
| 160/ | 0000; | 161/ | 0000; | 162/ | 0000; | 163/ | 0000 |
| 164/ | 0000; | 165/ | 0000; | 166/ | 0000; | 167/ | 0000 |
| 168/ | 0010; | 169/ | 1010; | 170/ | 1010; | 171/ | 0110 |
| 172/ | 1010; | 173/ | 0110; | 174/ | 0110; | 175/ | 1110 |
| 176/ | 0000; | 177/ | 0000; | 178/ | 0000; | 179/ | 0000 |
| 180/ | 0000; | 181/ | 0000; | 182/ | 0000; | 183/ | 0000 |
| 184/ | 0011; | 185/ | 1011; | 186/ | 1011; | 187/ | 0111 |
| 188/ | 1011; | 189/ | 0111; | 190/ | 0111; | 191/ | 1111 |
| 192/ | 0000; | 193/ | 0000; | 194/ | 0000; | 195/ | 0000 |
| 196/ | 0000; | 197/ | 0000; | 198/ | 0000; | 199/ | 0000 |
| 200/ | 1010; | 201/ | 0010; | 202/ | 0110; | 203/ | 1010 |
| 204/ | 0110; | 205/ | 1010; | 206/ | 1110; | 207/ | 0110 |
| 208/ | 1011; | 209/ | 0010; | 210/ | 0111; | 211/ | 1011 |
| 212/ | 0111; | 213/ | 1011; | 214/ | 1111; | 215/ | 0111 |
| 216/ | 1001; | 217/ | 0001; | 218/ | 0101; | 219/ | 1001 |
| 220/ | 0101; | 221/ | 1001; | 222/ | 1101; | 223/ | 0101 |
| 224/ | 0000; | 225/ | 0000; | 226/ | 0000; | 227/ | 0000 |
| 228/ | 0000; | 229/ | 0000; | 230/ | 0000; | 231/ | 0000 |
| 232/ | 1011; | 233/ | 0011; | 234/ | 0111; | 235/ | 1011 |
| 236/ | 0111; | 237/ | 1011; | 238/ | 1111; | 239/ | 0111 |
| 240/ | 0000; | 241/ | 0000; | 242/ | 0000; | 243/ | 0000 |
| 244/ | 0000; | 245/ | 0000; | 246/ | 0000; | 247/ | 0000 |
| 248/ | 1010; | 249/ | 0010; | 250/ | 0110; | 251/ | 1010 |
| 252/ | 0110; | 253/ | 1010; | 254/ | 1110; | 255/ | 0110 |

612

600

1,444,141

600

ROM #2

| | | | | | | | |
|------|-------|------|-------|------|-------|------|------|
| / | 0000; | 1/ | 0000; | 2/ | 0000; | 3/ | 0000 |
| 4/ | 0000; | 5/ | 0000; | 6/ | 0000; | 7/ | 0000 |
| 8/ | 0000; | 9/ | 0000; | 10/ | 0000; | 11/ | 0000 |
| 12/ | 0000; | 13/ | 0000; | 14/ | 0000; | 15/ | 0000 |
| 16/ | 0000; | 17/ | 0000; | 18/ | 0000; | 19/ | 0000 |
| 20/ | 0000; | 21/ | 0000; | 22/ | 0000; | 23/ | 0000 |
| 24/ | 0000; | 25/ | 0000; | 26/ | 0000; | 27/ | 0000 |
| 28/ | 0000; | 29/ | 0000; | 30/ | 0000; | 31/ | 0000 |
| 32/ | 0000; | 33/ | 0000; | 34/ | 0000; | 35/ | 0000 |
| 36/ | 0000; | 37/ | 0000; | 38/ | 0000; | 39/ | 0000 |
| 40/ | 0000; | 41/ | 0000; | 42/ | 0000; | 43/ | 0000 |
| 44/ | 0000; | 45/ | 0000; | 46/ | 0000; | 47/ | 0000 |
| 48/ | 0000; | 49/ | 0000; | 50/ | 0000; | 51/ | 0000 |
| 52/ | 0000; | 53/ | 0000; | 54/ | 0000; | 55/ | 0000 |
| 56/ | 0000; | 57/ | 0000; | 58/ | 0000; | 59/ | 0000 |
| 60/ | 0000; | 61/ | 0000; | 62/ | 0000; | 63/ | 0000 |
| 64/ | 0000; | 65/ | 0000; | 66/ | 0000; | 67/ | 0000 |
| 68/ | 0000; | 69/ | 0000; | 70/ | 0000; | 71/ | 0000 |
| 72/ | 0000; | 73/ | 0000; | 74/ | 0000; | 75/ | 0000 |
| 76/ | 0000; | 77/ | 0000; | 78/ | 0000; | 79/ | 0000 |
| 80/ | 0000; | 81/ | 0000; | 82/ | 0000; | 83/ | 0000 |
| 84/ | 0000; | 85/ | 0000; | 86/ | 0000; | 87/ | 0000 |
| 88/ | 0000; | 89/ | 0000; | 90/ | 0000; | 91/ | 0000 |
| 92/ | 0000; | 93/ | 0000; | 94/ | 0000; | 95/ | 0000 |
| 96/ | 0000; | 97/ | 0000; | 98/ | 0000; | 99/ | 0000 |
| 100/ | 0000; | 101/ | 0000; | 102/ | 0000; | 103/ | 0000 |
| 104/ | 0000; | 105/ | 0000; | 106/ | 0000; | 107/ | 0000 |
| 108/ | 0000; | 109/ | 0000; | 110/ | 0000; | 111/ | 0000 |
| 112/ | 0000; | 113/ | 0000; | 114/ | 0000; | 115/ | 0000 |
| 116/ | 0000; | 117/ | 0000; | 118/ | 0000; | 119/ | 0000 |
| 120/ | 0000; | 121/ | 0000; | 122/ | 0000; | 123/ | 0000 |
| 124/ | 0000; | 125/ | 0000; | 126/ | 0000; | 127/ | 0000 |
| 128/ | 1111; | 129/ | 1101; | 130/ | 1011; | 131/ | 1001 |
| 132/ | 0111; | 133/ | 0000; | 134/ | 0000; | 135/ | 0000 |
| 136/ | 1101; | 137/ | 1011; | 138/ | 1001; | 139/ | 0111 |
| 140/ | 1110; | 141/ | 0000; | 142/ | 0000; | 143/ | 0000 |
| 144/ | 1011; | 145/ | 1001; | 146/ | 0111; | 147/ | 1110 |
| 148/ | 1100; | 149/ | 0000; | 150/ | 0000; | 151/ | 0000 |
| 152/ | 1001; | 153/ | 0111; | 154/ | 1110; | 155/ | 1100 |
| 156/ | 1010; | 157/ | 0000; | 158/ | 0000; | 159/ | 0000 |
| 160/ | 0111; | 161/ | 1110; | 162/ | 1100; | 163/ | 1010 |
| 164/ | 1000; | 165/ | 0000; | 166/ | 0000; | 167/ | 0000 |
| 168/ | 0000; | 169/ | 0000; | 170/ | 0000; | 171/ | 0000 |
| 172/ | 0000; | 173/ | 0000; | 174/ | 0000; | 175/ | 0000 |
| 176/ | 0000; | 177/ | 0000; | 178/ | 0000; | 179/ | 0000 |
| 180/ | 0000; | 181/ | 0000; | 182/ | 0000; | 183/ | 0000 |
| 184/ | 0000; | 185/ | 0000; | 186/ | 0000; | 187/ | 0000 |
| 188/ | 0000; | 189/ | 0000; | 190/ | 0000; | 191/ | 0000 |
| 192/ | 1101; | 193/ | 1011; | 194/ | 1001; | 195/ | 0111 |
| 196/ | 1110; | 197/ | 0000; | 198/ | 0000; | 199/ | 0000 |
| 200/ | 1011; | 201/ | 1001; | 202/ | 0111; | 203/ | 1110 |
| 204/ | 1100; | 205/ | 0000; | 206/ | 0000; | 207/ | 0000 |
| 208/ | 1001; | 209/ | 0111; | 210/ | 1110; | 211/ | 1100 |
| 212/ | 1010; | 213/ | 0000; | 214/ | 0000; | 215/ | 0000 |
| 216/ | 0111; | 217/ | 1110; | 218/ | 1100; | 219/ | 1010 |
| 220/ | 1000; | 221/ | 0000; | 222/ | 0000; | 223/ | 0000 |
| 224/ | 1110; | 225/ | 1100; | 226/ | 1010; | 227/ | 1000 |
| 228/ | 0110; | 229/ | 0000; | 230/ | 0000; | 231/ | 0000 |
| 232/ | 0000; | 233/ | 0000; | 234/ | 0000; | 235/ | 0000 |
| 236/ | 0000; | 237/ | 0000; | 238/ | 0000; | 239/ | 0000 |
| 240/ | 0000; | 241/ | 0000; | 242/ | 0000; | 243/ | 0000 |
| 244/ | 0000; | 245/ | 0000; | 246/ | 0000; | 247/ | 0000 |
| 248/ | 0000; | 249/ | 0000; | 250/ | 0000; | 251/ | 0000 |
| 252/ | 0000; | 253/ | 0000; | 254/ | 0000; | 255/ | 0000 |

MEMORY UNIT

The calculator uses an all semiconductor memory system. Peripheral circuitry is bipolar and the memory consists of n-channel MOS read-only memory (ROM) and p-channel MOS read/write memory (RWM).

Addressing and physical layout of the memory module is done so that the number of words can be increased from 5K in the basic machine to 9K in the largest machine. The smallest increment of memory that can be added is 512 words.

The basic machine contains 9.5K words of memory, organized into 7.5K $\times$ 16 ROM, and 2K $\times$ 16 RWM. The 16-bit RWM words are divided into user registers and processor words.

The largest machine contains 15.5K words of ROM and 4K words of RWM.

Read/Write Memory

As shown in Figures 35—37B memory is made up of 1024 $\times$ 1, dynamics, read/write memory chips (Intel 1103). These devices are P-channel, MOS using silicon gate technology. To maintain the contents of memory, the device must be refreshed every 2 ms. This is accomplished by performing a read cycle at a given address. On each chip are 32 refresh amplifiers so that each read cycle, 32 cells get refreshed. The entire chip is then refreshed by cycling through the lower 5 address bits and reading each distinct address. The refresh period is 20 μ s at least every 2 ms.

Logic levels on all input lines to the RWM chips are 0 to + 16v. This includes the 3 clock lines (chip select, Y-enable or write, and precharge), 10 address lines, and input data. The output data, however, is a current of 600 μ a or more into 1K ohms or less. This low level output is "wire-or able" with other chips to build larger systems.

Read Only Memory

As shown in Figures 35 and 38—40 ROM chips are 4096 bit, n-channel MOS arranged 512 $\times$ 8. The devices are static and consume no power when not enabled. Data is retrieved from the ROMs by pulling the chip enable line from 0 to + 12v (turning the chip on), addressing the desired cells (0 or 4v levels) and selecting which output devices are to be enabled (4v or 0v). The output levels are sufficient to drive one TTL gate directly, and can be "wire-or/ed" for large systems.

As further shown in Figures 41 and 42A—D each ROM chip comprises six input buffers. These input buffers generate both the input and its complement. On the basis of the 64 possible combinations of the 6-inputs I_0 — I_5 , one of the 64 lines in the decoder is selected. The selected line enables one of the vertical lines in the 64 $\times$ 64 bit storage array. For example, let I_0 — I_5 =0 and I_6 — I_8 be "don't cares". This means line X00 (octal) is selected.

The two 8 out of 32 select decoders must choose 16 lines from the 64 horizontal lines selected by the vertical line X00. (The 8 out of 32 select decoder is actually a 2 out of 8 decoder repeated 4 times in each of the sections A—B). The output from four MOS Fet's a, b, c, and d are "wire or/ed." MOS devices a', b', c', and d' are also connected similarly. If I_6 and I_7 =0, horizontal lines 1XX, 2XX, 3XX, 5XX, 6XX, 7XX are grounded in each of the four sections A—B. This insures that MOS FET's b, c, d, b', c', and d' are non-conductive. This allows signals on lines 0XX and 4XX to pass into the output sections through transistors a and a'.

The output section contains the output buffer, 1 of 2 decoder, and the output drivers s. The output buffer provides a stage of gain and "wire or's" 4 lines from the storage array. The 1 of 2 decoder clamps the gates of 2 of the 4 output drivers in each section A—B by enabling either line I_6 or its complement ($\bar{I}_6$). This disables 1 of 2 signals coming from the output buffer. The output drivers then can be tied together with line (e) for a 512 $\times$ 8 organization.

Each of the above-listed constants and routines and subroutines of basic instructions employed by the calculator is stored in these ROM chips. The sixteen bits of each constant and basic instruction are stored in the 512₁₀ $\times$ 8₁₀ ROM chips by organizing the ROM chips into 64 $\times$ 64 bit matrices and computing the row and column numbers of each bit of each matrix by operating on each address and the particular bit (15 through 8, or 7 through 0). The column number is computed by subtracting the last two digits of the address from 100₈. For example, the column number of address 000=100₈—00₈=100=64₁₀ and the column number of address 777=100₈—77₈=1. The computation of the row number (referred to as IR in the

6A

602

1,444,141

602

flowchart of Figure 44) can best be described by referring to the flowchart of Figure 44 and the associated table of Figure 45. Once the row and column numbers are found it is a simple matter of storing in that location of the matrix that particular bit (i.e., a "1" or a "0"). A "0" is stored at a designated location by forming a metal gate to complete a MOS FET device at that location, and a "1" is stored at a designated location by leaving off the metal gate so that a MOS FET device is not formed at that location.

M-Register

As shown in Figures 35 and 46 A—B included on the M-Register board is the 16-bit Address or M-Register, all chip enable decoding and buffering, and address buffers for both ROM and RWM. The register uses four, four bit, serial in and out, parallel in and out shift Registers. Upon receipt of a TTM instruction from the microprocessor, serial data from the T-Bus is accepted into the M-register. Nothing is done with this data until either a read or write instruction is received, then one of two decoders are enabled. These chip Enable decoders uniquely decode which block of 512 words, either ROM or RWM, is being addressed. If ROM is being addressed, the signal is inverted and amplified to +12v. For RWM the Chip Enable enables a gate, which allows a 16 Volt clock signal to reach the enabled RWM chips. The clock wave-form is generated on the control card.

The dynamic characteristic of the RWM chips, requires that all chips be enabled simultaneously during a refresh cycle, to refresh the entire read/write memory. The buffer circuits in the output of the Chip enable decoders allow the chip select clock to reach all of the RWM chips during refresh but only those being accessed, during a read or write cycle.

Totem Pole outputs or gates with resistor pull-ups are used as buffers for the ROM address lines. Using the totem pole output gates, the effects of crosstalk can be minimized while the resistor pullup lifts the address lines above the required 4v level. The nand gates are enabled during a memory cycle so that the ROM address lines are inhibited at a 5v level. The RWM address lines must pull from 0v to + 16v. High voltage, open collector, inverters with discrete transistor pull-ups are used as buffers for all the address bits.

Control

A memory cycle consists of a read or write instruction from the processor accompanied by 12 clock pulses from the shift clock. As shown in Figures 35, 46 A—B and 74A—B, control uses these pulses and instructions to generate the clocks required by the RWM chips. A synchronous system of flip-flops and gates are used. The outputs from the flip-flops are then buffered to become the required clock signals (Pre-charge, Y-enable, chip select).

Refreshing the read/write memory is also taken care of by the control. An astable multivibrator with a repetition rate of 500 HZ minimum generates a signal which allows a refresh cycle to occur. A flip-flop generates the actual signal (REF), but only if the astable multivibrator signal is high, there is no read or write cycle in progress and the processor signal, CCT, is high. CCT goes high between processor instructions, thus it is known that nothing is going to be interrupted when REF is generated. REF is then buffered by an open collector inverter and given to the processor INH. INH halts the machine and the refresh cycle begins.

The same system used for a memory cycle is used during refresh to again generate the necessary clocks (Precharge and chip select). When the system returns to state 0 and REF is present, a counter is advanced one count. This counter provides the refresh addresses which go to the RWM only if REF is present. When this counter returns to state 0, it causes REF and INH to return to preset conditions and the machine continues normal operation.

Another function of the control is to provide for extended memory capability. The control handles any external memory as if it were an extension of the internal memory. From the user's point of view, he does not need to know if an extended memory is connected other than the fact that available memory has increased.

In addition, the control has the provision for extracting information from or loading information into the calculator T-Register through the D-Bus (data bus).

Other signals generated on the control are employed to direct the flow of data in the T-register.

T-Register

Data to and from the memory is temporarily stored in the T-register. As

shown in Figures 35 and 48A—B four 4-bit, serial in and out, parallel in and out shift registers make up the actual T-register. The registers have a mode control (TMC) which when low, allows serial data flow and when high, allows parallel data flow.

Serial data enters the T-register in the presence of the $\overline{\text{TTS}}$ instruction, and in the presence of a $\overline{\text{TTS}}$ recirculated in the T-register to prevent loss of data.

Parallel data is accepted from either ROM or RWM during a read cycle. The ROM data is buffered by NAND gates and the RWM by sense amplifiers followed by the same NAND gates. All 16 bits are read from the ROM simultaneously. Eight bits are read from RWM twice during a read cycle. The eight bits to be written into RWM have their own discrete buffer stage that translates T²L logic levels into 16v logic levels used by the RWM.

MEMORY SYSTEM MNEMONIC TABLE

SIGNALS GENERATED OUTSIDE MEMORY I/O CONNECTOR

| | | | |
|----|---------------------------|--|----|
| 15 | $\overline{\text{CCT}}$ — | Control clock-not, the inverted envelop of SCK. | 15 |
| | SCK — | Shift clock. | |
| | MCK — | Memory clock, a continuous pulse train, used by the memory control for timing of the memory and refresh cycles. | |
| | IOD — | I/O Data. Goes to control board to be gated to $\overline{\text{S-BUS}}$. | |
| 20 | ITS — | I/O to $\overline{\text{S-BUS}}$, the signal which gates IOD to $\overline{\text{S-BUS}}$. | 20 |
| | SCO & —
SCI | Coded signals which generate
UTS — Units to $\overline{\text{S-BUS}}$
ZTS — Zero to $\overline{\text{S-BUS}}$
MTS — M-Reg to $\overline{\text{S-BUS}}$
TTS — T-Reg to $\overline{\text{S-BUS}}$ | |
| 25 | $\overline{\text{TTS}}$ — | T—BUS to T-Reg, OV=True. | 25 |
| | T—BUS | Data on this bus acts as inputs to M & T registers. | |
| | $\overline{\text{RDM}}$ — | Read memory, negative true. Lasts for 12 clock pulses. | |
| | $\overline{\text{WTM}}$ — | Write memory, negative true. Lasts for 12 clock pulses. | |
| 30 | $\overline{\text{INH}}$ — | Inhibit, negative true. The processor is stopped whenever $\overline{\text{INH}}$ is at zero volts. The memory control generates this signal while a R/W memory refresh cycle is present. I/O also generates it. | 30 |

OTHER SIGNALS AT I/O CONNECTOR

| | Name | Source | |
|----|---------------------------|---|----|
| 35 | T00 | T-Register | |
| | T01 | " " | 35 |
| | T02 | " " | |
| | T03 | " " | |
| | D-BUS — | Data Bus — external data (extended memory data) enters machine via this bus. | |
| 40 | $\overline{\text{EDT}}$ — | External Data Transfer gates D-Bus data into machine OV=True. | 40 |
| | $\overline{\text{EMB}}$ — | Extended memory busy. Signal provided by extended memory that tells memory control. | |

- a. Extended memory cycle is complete
- b. Extended memory is present.

SIGNALS GENERATED ON READ/WRITE MEMORY CARDS

RWD(XX) — Read/Write data. Output from the 1103 memory.

600 μ A into 150 Ω ~ 1
0 current ~ 0

OTHER SIGNALS USED BY (RWM)

| <u>Name</u> | <u>Source</u> |
|-------------|---------------|
| A00-A04 | CONTROL |
| A05-A09 | M-REG |
| CEN | M-REG |
| RWI(XX) | T-REG |
| R/W | CONTROL |
| PCG | CONTROL |

SIGNALS GENERATED ON T-REGISTER CARD

T00-T15 — T-register data bits. Used as data into memory. T00-T03 are also outputs to the CPU. (4 bit processing).

RWI(XX) — Read/Write inputs. T-register data gates to Read/write memory. +16V-GND.

OTHER SIGNALS USED BY T-REGISTER

| <u>Name</u> | <u>Source</u> |
|-------------|---------------|
| ROD(XX) | ROM |
| TRI | CONTROL |
| TSC | CONTROL |
| TMC | CONTROL |
| TPC | CONTROL |
| RWD(XX) | R/W MEM |
| RWE | M-REG |

SIGNALS GENERATED ON M-REGISTER CARD

M00-M15 — M-register data bits. Used to generate address and chip select information. (M00 also is gated out on S-BUS by MTS).

I00-I07 — ROM address bits. Decodes down to two bits available at ROM output buffers.

I08-I08 — Selects which ROM output buffers are enabled.

617

605

1,444,141

605

SIGNALS GENERATED ON M-REGISTER CARD (Continued)

| | | | |
|---|-----------|--|---|
| | CS(XX) — | Chip enable, basic machine selects which ROM chips are turned on (+12V—ON) | |
| | AEN — | Address enable. $AEN = RDM + WTM$ | |
| 5 | A05-A09 — | Address bits for R/W memory. (+16V & GND) | 5 |
| | CEN — | Chip select, basic machine a negative true clock which selects which R/W chips are turned on. (+16V & GND) | |
| | RWE — | Read/Write enable. A +5V signal any time a R/W chip is addressed for a machine memory cycle. | |

10

OTHER SIGNALS USED BY THE M-REGISTER

10

15

| Name | Source |
|-------|-----------|
| T-BUS | PROCESSOR |
| SCK | " |
| RDM | " |
| WTM | " |
| TTM | " |
| VLD | CONTROL |
| MTS | " |
| REF | " |

15

20

SIGNALS GENERATED BY CONTROL

20

| | | | |
|----|-----------|--|----|
| | VOR — | A signal generated half way through the memory cycle to disable the active pull up devices on the ROM outputs. | |
| | TRI — | T-Register input $TRI = (T-BUS) \cdot (TTT) + (T00) \cdot (TTS)$ | |
| | MTS — | M-register to $\bar{S}$ -BUS. Generated from SC0, $\bar{SCI}$ | |
| 25 | A00-A04 — | Address bits for R/W memory also used during memory refresh. | 25 |
| | TSC — | T-Register series clock. $TSC = SCK \cdot (TTS + TTT + EDT)$ | |
| | TPC — | T-Register parallel clock. (Strobes in data from memory) only only during internal memory read cycle. | |
| 30 | R/W — | Read/Write. A clock which left at +16V for a read and clocked to GND during a write. (R/W memory only) | 30 |
| | PCG — | Precharge. The 3rd 16V clock required by the 1103 R/W memory chips. | |
| | REF — | Refresh. OV when the memory is in a refresh cycle. | |

SIGNALS GENERATED BY CONTROL (Continued)

| | | | |
|----|----------------------|--|----|
| 5 | $\overline{CEM}$ — | Call extended memory. Prevents ROM clock from changing μ processor states. Given for all read and write commands. Signal is removed if the memory cycle is not extended memory cycle. If extended memory cycle, $\overline{CEM}$ is removed after extended memory has completed cycle. $0^v = \text{True}$. | 5 |
| | $\overline{S-BUS}$ — | Gates I/O register, data, $T00$, $M00$ or ones onto $\overline{S-BUS}$ and sent to processor. $0 = \text{True}$. | |
| 10 | $\overline{INH}$ — | Inhibit, negative true. The processor is stopped whenever $\overline{INH}$ is at zero volts. The memory control generates this signal while a R/W memory refresh cycle is present. I/O also generates it. | 10 |
| | $\overline{TMC}$ — | T-reg mode control. $\overline{TMC}=0$; T-reg is set up to accept information from memory. $\overline{TMC}=1$; T-reg. is set up to shift serially. | |
| 15 | $\overline{EMC}$ — | Extended memory cycle. $+5^v$ signal used to signal extended memory to begin its cycle. $0^v = \text{True}$. | 15 |
| | $\overline{VOR}$ — | A signal generated half way thru memory cycle to allow data to flow out of ROM. | |

OTHER SIGNALS USED BY THE CONTROL

| | Name | Source | |
|----|----------------------|-----------------|----|
| 20 | $\overline{CCT}$ | PROCESSOR | 20 |
| | $\overline{TOO}$ | T-REG | |
| | $\overline{IOD}$ | PROCESSOR | |
| | $\overline{ITS}$ | " | |
| | $\overline{SCO}$ | " | |
| 25 | $\overline{SCI}$ | " | 25 |
| | $\overline{TTT}$ | " | |
| | $\overline{T-BUS}$ | " | |
| | $\overline{SCK}$ | " | |
| | $\overline{RDM}$ | " | |
| 30 | $\overline{WTM}$ | " | 30 |
| | $\overline{MCK}$ | " | |
| | $\overline{M00-M04}$ | M-REG | |
| | $\overline{AEN}$ | M-REG | |
| | $\overline{EDT}$ | EXTENDED MEMORY | |
| 35 | $\overline{D-BUS}$ | I/O CONNECTOR | 35 |
| | $\overline{EMB}$ | EXTENDED MEMORY | |

SIGNALS GENERATED ON ROM BOARD

ROD(XX) — Read Only Data

OTHER SIGNALS USED BY ROM

| <u>Name</u> | <u>Source</u> |
|-------------|---------------|
| CS(XX) | M-REG |
| 100—107 | M-REG |
| 108—108 | M-REG |
| VOR | CONTROL |

INPUT-OUTPUT CONTROL UNIT

The input-output control unit allows the calculator to communicate with the internal input, input-output, and output units and with external peripheral devices. As shown in Figures 31 A—C and 49 A—D, the input-output control unit is contained on two printed circuit boards, the "control and system clock" board and the "I/O register and gate interface" board. A third board, shown in Figure 50, is an I/O motherboard providing room for connecting four external interface cards to the calculator.

The internal input, input-output, and output units are distinguished from peripheral devices by the fact that the I/O language set addresses them directly. Hence, each I/O instruction contains an internal peripheral address as part of its makeup. The four internal directly-addressable input, input-output, and output units are the I/O register, the magnetic card reading and recording unit, the output printer unit and display unit.

The external peripheral devices are indirectly addressable and are connected via cable to an interface card which is plunged into the I/O motherboard at the rear of the calculator. The term indirectly addressable is defined here to mean the external peripheral devices are addressed by lines leading from the four most significant bits in the I/O register, thereby requiring an address word to be loaded into the directly addressable I/O register.

I/O CONTROL AND SYSTEM CLOCK SECTION

The function of the I/O control and system clock section is to provide control to the I/O register and gate interface section. This is accomplished by use of an I/O instruction set stored in the main memory of the calculator.

The microprocessor causes instructions from the memory unit to be loaded into the T-Register and then to be transferred to the Q-Register. The microprocessor determines the type of instruction and causes the proper execution of the instruction. If the instruction is an I/O type, control is transferred by the microprocessor to the I/O control and system clock section.

The microprocessor remains in a two-state waiting loop while the I/O control section is active. Time in the wait loop is between .72 micro seconds and 6.5 micro seconds.

Bits 5 through 10 from the Q-Register are connected to the I/O control section and remain constant during an I/O instruction execution time. Bits 5 through 8 representing the I/O instruction code are gated to the I/O address flip flops and entered on each clock time while the I/O is inactive. The four outputs of the address flip flops are connected to the address input of a 1 of 16 decoder and represent the starting state address of the I/O instruction to be executed. When the I/O control section is enabled, the input gates passing bits 5 through 8 to the I/O address flip flops are closed and the 1 of 16 decoder enabled. This allows the starting state I/O micro instructions to come from the 1 of 16 decoder. The next state address coming from the closed input gates will be the exit state (1111=17) unless modified by reopening the gates to let the original starting state code through or by modifying the output of one or more of the input gates using a "wire or" connection coming from the 1 of 16 decoder output. This address is sent to the I/O address flip flops inputs and clocked in on the leading edge of the first half

clock cycle. The first half clock cycle turns off the 1 of 16 decoder and the address changes. The second half clock cycle enables the 1 of 16 decoder, allowing the next state micro instruction to appear. (See Figure 51 for the timing described above). This process continues until the exit state is encountered. On the exit state, the I/O Control is disabled and control is returned to the microprocessor.

The I/O instructions involving the transfer of data between the I/O and the CPU (OT, LI, MI), require 16 passes through the same state (1 pass for each of 16 bits). This is achieved by checking the output of a 16-bit down counter and then decrementing after each pass through the state. If the counter indicates "0" has not been reached, it causes the starting state address to be reloaded into the address flip flops by opening the input gates. When 16 passes have been indicated by the counter, the input gates are not allowed to open; however, the next state (1111) is modified by the output of the 1 of 16 decoder through a "wire or" connection on the 2nd bit to give state 1101. This address is input to the I/O address flip flops as in the preceding paragraph.

The above-described operation of the I/O control section is also illustrated and further described in the flow chart of Figure 52.

Bit 9 is called a hold/clear bit. It allows a clear flag (CLF) to take place or not to take place after execution of the other I/O instructions (STF excepted).

Bit 10 is used in conjunction with the micro instructions PTR and XTR to give control to the I/O.

The I/O control and programmable clock mnemonics are given in the following table:

I/O CONTROL BOARD MNEMONICS

| | | | |
|----|------------------|---------------------------|----|
| 25 | $\overline{CC0}$ | Clock Code Zero | 25 |
| | CC1 | " " One | |
| | CC2 | " " Two | |
| | CC4 | " " Four | |
| | CC8 | " " Eight | |
| 30 | CCT | Control Clock to Tester | 30 |
| | CEM | Call Extended Memory | |
| | $\overline{CLC}$ | Clear Control | |
| | CLF | Clear Flag | |
| | DRC | Data Register Clock | |
| 35 | EBT | Eight Bit Transfer | 35 |
| | EOW | End of Word | |
| | IIO | Inhibit Internal OSC | |
| | INH | Inhibit Clock | |
| | IPS | Inhibit Primary/Secondary | |
| 40 | ITS | Input to S-Bus | 40 |
| | MCK | Memory Clock | |
| | $\overline{POP}$ | Power On Pulse | |
| | PTR | P-Reg to R-Bus | |

I/O CONTROL BOARD MNEMONICS (Continued)

| | | | |
|----|-------------------------|-----------------------------|----|
| | QFG | Qualifier Flag | |
| | Q5 | „ Five | |
| | Q6 | „ Six | |
| 5 | Q7 | „ Seven | 5 |
| | Q8 | „ Eight | |
| | Q9 | „ Nine | |
| | Q10 | „ Ten | |
| | QRD | „ ROM Disable | |
| 10 | RCA | ROM Clock Address | 10 |
| | RCF | ROM Clock Flip Flop | |
| | SCB | Set Carry Bit | |
| | SCK | Shift Clock | |
| | SCT | „ „ to Tester | |
| 15 | $\overline{\text{SRA}}$ | Service Request Acknowledge | 15 |
| | STC | Set Control | |
| | STF | Set Flag | |
| | TCK | Tester Clock | |
| | TTO | T-Bus to Output | |
| 20 | $\overline{\text{TTX}}$ | T-Bus to A/B Reg. | 20 |
| | XTO | External OSC | |
| | $\overline{\text{XTR}}$ | A/B Reg. to R-Bus | |

Note:

(—) indicates negative true signal

I/O REGISTER AND GATE INTERFACE SECTION

As shown in Figures 49 A—D, the directly addressable I/O register (address 01) is a 16 bit universal parallel in/out, serial in/out register that is connected to the calculator processor by the serial-in S-Bus and the serial-out T-Bus. Information is passed non-inverted from the A or B registers bit serial to the I/O register with the I/O instruction OXT 01. Sixteen lines connected to the parallel outputs of the I/O register provide data out to the internal input, input-output, and output units and to the external output interfaces. (NOTE: each I/O unit or interface may place only 1 TTL load on the output lines.)

Parallel entry to the I/O register is through 12 party lines connected to the 12 least significant parallel inputs. The input lines are negative true with all input interfaces tying to the lines through open collectors. Care must be taken to insure there is no disturbance to the lines while an interface is inactive. Input information is passed inverted to the A or B register bit serially with the I/O instructions LIX 01 or MIX 01. (The inversion puts positive true information into the A or B register).

Input information is entered into the I/O register in three ways:

a) Service Request.

Entry by the service request method is controlled by a service inhibit flip flop. When the service inhibit flip flop has been cleared with the I/O instruction CLF 01, a service request may be initiated by returning the SSI (Service Strobe Input) party line to ground through an open collector on the interface. This signal causes the parallel inputs to be strobed into the I/O register and sends a request for service (QNR) to the microprocessor. The microprocessor prior to receiving a request for service would have been cycling through various instruction paths and checking for a service request after execution of each instruction. Upon receipt of a request for service, the processor interrupts the sequence of instructions it was doing and loads an address into the M-Register which contains the starting address of the service routine. At the same time a signal, SRA (Service Request Acknowledge), turns off the service inhibit flip flop and also sets the single service flip flop which permits only one service interrupt to the processor per service strobe input. The single service flip flop is reset when the service strobe is removed. All lines from an interface using the service request method for entering information are inhibited when the service inhibit flip flop is set.

b) Return of Channel Flag After Command is Given to an External Peripheral Device.

This method implies the calculator must control the peripheral. That is to say the calculator transmits the indirect address and control enable (CEO) from the "I/O Register and gate interface" section to the interface with the expectation of information being returned by the peripheral through the interface to the I/O register. Because of this expectation, only limited instructions may be performed by the calculator while waiting. The service request method must be inhibited during this wait so that input information is not destroyed by another peripheral using service request. When a controlled peripheral responds, its flag and data are processed at the interface. The signal CFI (Channel Flag In) causes the loading of parallel data from the interface into the I/O register and clears the control enable flip flop so that the CEO signal is removed from the interface. The calculator can interrogate the control enable flip flop with the instructions SFS 01 or SFC 01 to determine when data has been loaded in.

c) Giving the I/O Instruction STF 01.

The instruction STF 01 as described in (a) sets the service inhibit flip flop inhibiting the service request mode of entry. The STF 01 instruction also causes a parallel load of the input lines into the I/O register.

The output display (address 08) receives information from the I/O register. A 16 bit word is transferred to the I/O register with the instruction OTX 08. The address 08 allows the display enable flip flop to be set with the micro-instruction EOW after the 16th bit has been transferred. The display enable flip flop sends a signal DEN to the display indicating information is ready in the I/O Register. The display enable flip flop is cleared with the I/O instruction CLF 08.

The keyboard operates as described below. 7 bit ASCII assigned keycodes are entered into the calculator by an interrupt process. When a key on the keyboard is depressed the keyboard interface card requests service. Input data is stored along with the request for service on the keyboard interface card. The stored signal for service is gated with the Prevent Interrupt signal through an open collector NAND gate onto the Service Request party line (SSI = Low for service). The giving of Service Request causes the I/O register to be loaded. However, input data from the keyboard interface card is not enabled yet. Thus all status and data inputs are high. This indicates to the CPU that a keyboard is interrupting. An OT x 16 instruction is given by the firmware. The select code of 16 enables the gate of the data input lines by a STF 1 instruction and data is loaded into the I/O register. LIA 0 allows data to be taken from the I/O register.

All external peripheral interfaces are indirectly addressed from the four most significant bits in the I/O register. Thus to communicate with an external peripheral, an address (0000 excluded) must be loaded into the I/O register. Data and status will be loaded at the same time if the peripheral is to act as a receiver. If the peripheral is to act as a transmitter, only the address and status need be loaded. Next, the I/O instruction STC 01 sets the Control Enable Out flip flop. This flip

628

611

1,444,141

611

5 flop sends a signal $\overline{CEO}$ to all external interface slots. The $\overline{CEO}$ signal and the decoded (from the 4 bit address) address allows the interface to command the peripheral. After the peripheral has responded, information given back to the interface by the peripheral is processed to the I/O register in the manner described above under (b) "Return of Channel Flag After Command is Given to an External Peripheral Device". 5

The I/O register and gating control circuit mnemonics are given in the following table:

I/O REGISTER AND GATE BOARD

| | | | |
|----|--------------------------|---------------------|----|
| 10 | $\overline{CEO}$ | Control Enable Out | 10 |
| | $\overline{CFI}$ | Channel Flag In | |
| | CLF | Clear Flag | |
| | CO0, 1, 2, 3 | Code Out | |
| | $\overline{DEN}$ | Display Enable | |
| 15 | DI0, 1, 2, 3, 4, 5, 6, 7 | Data In | 15 |
| | DO0, 1, 2, 3, 4, 5, 6, 7 | Data Out | |
| | DRC | Data Register Clock | |
| | EBT | Eight Bit Transfer | |
| | EOW | End of Word | |
| 20 | IOD | I/O Data | 20 |
| | KLS | Key Lights Strobe | |
| | MCR | Mag Card Reset | |
| | $\overline{MFL}$ | Mag Flag | |
| | MLS | Mag Latch Strobe | |
| 25 | PEN | Printer Enable | 25 |
| | $\overline{POP}$ | Power On Pulse | |
| | PTF | Printer Flag | |
| | Q0 | Qualifier Bit 0 | |
| | Q1 | " " 1 | |
| 30 | Q2 | " " 2 | 30 |
| | Q3 | " " 3 | |
| | Q4 | " " 4 | |
| | QFG | " Flag | |
| | QNR | " Not Request | |
| 35 | $\overline{SIH}$ | Service Inhibit | |
| | SI0, 1, 2, 3 | Status In | |

604

612

1,444,141

612

I/O REGISTER AND GATE BOARD (Continued)

| | | | |
|---|-------------------------|-----------------------------|---|
| | SO0, 1,2,3 | Status Out | |
| | $\overline{\text{SRA}}$ | Service Request Acknowledge | |
| | $\overline{\text{SSI}}$ | Service Strobe In | |
| 5 | STC | Set Control | 5 |
| | STP | Stop | |
| | STF | Set Flag | |
| | T-Bus | T-Bus | |
| | TTO | T-Bus to Output | |

10 NOTE: ($\overline{\quad}$) indicates negative true signal 10

15 As shown in Figure 53, when addressing a peripheral device, bits loaded into the 4 most significant locations in the I/O register from the CPU constitute the peripheral address code. As part of the output party line system the address code is routed to all I/O interface slots. Each I/O interface card decodes the 4 line address code to a unique single line for use on that particular I/O card. The binary codes 10 through 15 have been reserved for dedicated peripheral addresses which are used by dedicated keys (from the keyboard) and dedicated I/O drivers. Binary codes 1 through 9 are for general use. Code "0" is a non-addressing code and is used in operations that do not involve addressing a specific peripheral. The following table summarizes the address code assignments: 20

ADDRESS CODE ASSIGNMENTS

| ADDRESS | 4-BIT CODE | ASSIGNED PERIPHERAL |
|---------|------------|---|
| 15 | HHHH | TYPEWRITER |
| 14 | HHHL | PLOTTER |
| 13 | HHLH | |
| 12 | HHLL | KEYBOARD & KEYBOARD-LIKE PERIPHERALS |
| 11 | HLHH | |
| 10 | HLHL | |
| 9 | HLLH | GENERAL USE ; ONE OF NINE SELECTABLE |
| 8 | HLLL | " |
| 7 | LHHH | " |
| 6 | LHHL | " |
| 5 | LHLH | " |
| 4 | LHLL | " |
| 3 | LLHH | " |
| 2 | LLHL | " |
| 1 | LLLH | " |
| 0 | LLLL | USED ON INTERRUPT I/O INTERFACE CARDS
WHEN THE INTERRUPT BECOMES ENABLED |

The general usage codes (1—9) are decoded outputs from a 4 line to 1 of 10 decoder (SN 7442 for example). It is intended that the codes 1 through 9 be jumper selectable. This would allow the user to select a code for his system peripherals or allow him to use more than one of the same peripheral by selecting different address codes.

Since the I/O register is used to communicate with the internal input, input-



output, and output units as well as peripheral devices, a given peripheral's address code will appear randomly in the I/O register address field with there being no intention of expecting the peripheral to respond. Therefore, a second piece of information is necessary for the I/O interface card to form a unique signal which will indicate to the peripheral to respond. This second piece of information is control information and is described hereinafter:

The I/O interface cards contain TTL compatible logic for manipulating control and data from the calculator and/or the peripheral. All I/O interface cards which are intended to be used with the calculator must provide storage either on the I/O interface card or in the peripheral. Thus data being transferred from the calculator to the I/O card must be stored at the instant the peripheral is requested to respond. Likewise data coming from a peripheral must be stored until the calculator accepts it. This requirement is important and must be considered on all compatible interface cards.

The calculator can supply up to 100 ma. maximum at +5 volts to each I/O interface card. Power exceeding this absolute maximum must be supplied by the peripheral.

The following table lists the pin assignments for all I/O lines at the plug-in slots on the calculator back plane, as viewed from the rear of the calculator, left to right.

EXTERNAL I/O INTERFACE PIN ASSIGNMENTS

| | | | |
|----|------------------|---|------------------|
| 1 | $\bar{L}$ | A | $\bar{L}$ |
| 2 | +5 | B | +5 |
| 3 | USED | C | USED |
| 4 | USED | D | 10/20 |
| 5 | USED | E | USED |
| 6 | DI 0 | F | DO 0 |
| 7 | DO 1 | H | DO 2 |
| 8 | DI 3 | J | DO 3 |
| 9 | DI 2 | K | DI 1 |
| 10 | DO 4 | L | DI 4 |
| 11 | DO 5 | M | DI 5 |
| 12 | DO 6 | N | DI 6 |
| 13 | DO 7 | P | DI 7 |
| 14 | SO 0 | R | SI 0 |
| 15 | SO 1 | S | SI 1 |
| 16 | SO 2 | T | SI 2 |
| 17 | SO 3 | U | SI 3 |
| 18 | CO 0 | V | CO 1 |
| 19 | CO 2 | W | CO 3 |
| 20 | $\overline{SSI}$ | X | $\overline{SIH}$ |
| 21 | $\overline{CEO}$ | Y | $\overline{CFI}$ |
| 22 | $\bar{L}$ | Z | $\overline{STP}$ |

5
10

The chart below lists all I/O lines with brief definitions and specifications and Figure 50 shows the source and relative relationship of the I/O lines. The output address data lines (Co 0—3) transmit the address code along the party lines to all interface slots. These lines will go high and low according to information being shifted in or out of the I/O Register. At anytime a peripheral is addressed the lines will become steady 1 instruction time ($8 \mu s$) before control information is passed to the I/O interface card or before data or status is taken from the I/O interface card and will remain constant until the control information is removed. After the control information is removed, the state of the I/O lines become unpredictable until the next addressing takes place. Address data coming to the I/O interface card is positive true and each interface may place 1 TTL load on each address line.

5
10

I/O Line Specification Chart

| | Name of Line | Line Definition | Direction | Load/Loading | Voltage | | # of Lines |
|---|---------------------------------|--|-----------|--|--|---------------------------|------------|
| | | | | | High | Low | |
| 1 | Address Data
(CO 0-3) | Transmits a 4 bit address from the I/O Register to be recognized by an interface card. (Data - High) | Out | 1 TTL (1.6 ma) allowed per interface. | $\geq 2.4v$ | $\leq .4v$ | 4 |
| 2 | Device Ready
<u>CEO</u> | Indicates calculator is ready for information interchange with an addressed peripheral.
(Active State - Low) | Out | 1 TTL (1.6 ma) allowed per interface. | $\geq 2.4v$ | $\leq .4v$ | 1 |
| 3 | Device Request
(CFI) | Acknowledges receipt of data by a peripheral from the calculator or indicates data is to be input to the calculator.
(Active State - Low) | In | Loading of 6.6 ma to the interface card. | 1k resistor to + 5v. use open collector. | Must be driven below .4v. | 1 |
| 4 | Halt Status
(STP) | Indicates stop key has been depressed.
(Active State - Low) | Out | 1 TTL (1.6 ma) Allowed/interface. | $\geq 2.4v$ | $\leq .4$ | 1 |
| 5 | Input Data
(DI 0-7, SI 0-3) | Receives input data to I/O register.
(Data - Low) | In | Loading of 6.1 ma | 1k Res. To + 5v | Driven $\leq .4v$ | 12 |
| 6 | Output Data
(DO 0-7, SO 0-3) | Transmits Data from the I/O register.
(Data - High) | Out | 1 TTL (1.6 ma) Allowed/interface. | $\geq 2.4v$ | $\leq .4v$ | 12 |
| 7 | Prevent Interrupt
(SIH) | Indicates data cannot be entered under service request. (Interrupt)
(Active State - Low) | Out | 1 TTL (1.6 ma) Allowed/interface. | $\geq 2.4v$ | $\leq .4$ | 1 |
| 8 | Service Request
(Lo) (SSI) | Indicates a CPU interrupt is to take place to allow data to enter.
(Active State - Low) | In | Loading of 6.6 ma | 1k Res. to + 5v | Driven $\leq .4$ | 1 |

The output data lines (DO 0—7) output data from the A or B accumulator in 8 bit bytes from the 8 least significant locations in the I/O register to all interface card slots. The logic state is positive true (Data = 1 = High). Each interface card may place 1 standard TTL load on each data line.

The output data status lines (SO 0—3) output status data from the A or B accumulator and are driven from the next four locations above the data out positions in the I/O register. (DO positions = 0 to 7; SO positions = 8 to 11). These lines are used for sending additional information to a peripheral. The logic state is positive true. One standard TTL load may be placed on each output data status line. (Special drivers, fast data transfer, and interrupt do not make use of SO 3).

The input data lines (DI 0—7) transmit input data in 8 bit bytes to the 8 least significant bit positions of the I/O register (Locations 0 to 7) from the I/O interface card. Each "Data In" line has a 1K pull up resistor to +5 volts and under the party line system must be driven low for a logical 1 from open collector gates on each addressed I/O interface card. The logic state is negative true.

The input data status lines (SI 0—3) receive information from the I/O interface cards and transmit it to location 8 through 11 in the I/O register. Each line has a 1K pull up resistor to +5 volts. These lines are used to provide additional information to the calculator about the state of a peripheral. The logic state is negative true.

The negative true "Device Ready" output line ($\overline{\text{CEO}}$) transmits a control signal, which when combined with an address code will initiate a peripheral response on the addressed I/O interface card. "Device Ready" is controlled by the I/O interface driver and therefore may look different depending upon the driver. For example, when the calculator wishes to transmit data to the I/O interface card or to initiate a peripheral response prior to receiving data from the peripheral, the calculator causes the "Device Ready" output line to go low and stay low until the peripheral response is over and the calculator receives the signal "Device Request" ($\overline{\text{CFI}}$) from the I/O interface card. The "Device Ready" flip-flop always receives a clear signal whenever the I/O interface completes a parallel load.

The "Device Request" party line $\overline{\text{CFI}}$ when driven low from an open collector gate on the I/O interface card will cause the loading to parallel input information into the 12 least significant locations of the I/O register. The active state of the line is low (negative true).

The peripheral flag, indicating to the I/O interface card the peripheral has received data/control or is ready to input data, is gated through an open collector nand gate onto the "Device Request" ($\overline{\text{CFI}}$) party line. The open collector gate is enabled by the I/O interface card's address and "Device Ready" ($\overline{\text{CEO}}$). The "Device Request" line is pulled up inside the calculator by a 1K resistor to +5 volts.

The "Device Request" ($\overline{\text{CFI}}$) signal must stay low until "Device Ready" ($\overline{\text{CEO}}$) has been cleared (goes high). At this time data transfer has terminated and peripheral's flag and control must be cleared in preparation for the next pass. Since a parallel load in the I/O register causes the "Device Ready" flip-flop to receive a clear signal, when a "Device request" ($\overline{\text{CFI}}$) is entered, a parallel load takes place and afterwards "Device Ready" ($\overline{\text{CEO}}$) is cleared. The calculator uses "Device Request" in its general mode of data transfer.

The "Halt Status" output line ($\overline{\text{STP}}$) is a line that goes low when the STOP key on the calculator is depressed. It will stay low for the duration of the key depression. One standard TTL load may be placed on this line by each I/O interface card.

The "Prevent Interrupt" output line ($\overline{\text{SIH}}$), when low indicates to the I/O interface card that a request for service must not be given to the calculator. One standard TTL load may be placed on this line by each I/O interface card.

The "Service Request" (Lo) line ($\overline{\text{SSI}}$), when driven low causes the loading of parallel input information into the 12 least significant locations of the I/O register and causes a CPU interrupt for service. The peripheral's request for service is gated with the "Prevent Interrupt" ($\overline{\text{SIH}}$) line onto the "Service Request" party line through an open collector nand gate. A 1K pull-up resistor to +5 volts is connected to the line inside the calculator.

The general format for all data transfer consists of 8 bit parallel bytes. Other data formats are handled by specially developed drivers, such as the ROM plug-in module employed for driving the typewriter.

The state of a peripheral is generally checked before attempting an output. This is done by first inhibiting the interrupt system. The address of the I/O

interface card is shifted into the I/O register. The decoded address code enables the open collector gates on the I/O interface card. The status of the peripheral is passed to the "Status In" lines and loaded into the I/O register with an I/O instruction issued by the calculator. The I/O register information is transferred to the A or B accumulator and processed. If the peripheral is ready, the output data word consisting of the address code, output status (if necessary) and the eight bit data byte is formed in the A or B accumulator. The output data word is transferred to the I/O register after which the "Device Ready" (CEO) flip-flop is set. The I/O interface card receives the data, address code and "Device Ready" and a peripheral response is initiated. The calculator interrogates the state of the "Device Ready" flip-flop to determine when the I/O interface card has received the information and the peripheral response is done. The peripheral I/O interface card signals the calculator it is done by transmitting the "Device Request" (CFI) signal to the calculator. The output waveforms are shown in Figure 54.

Before inputting data from the I/O interface card it is necessary to determine if the peripheral has responded and is ready to input data. After a peripheral response has been initiated, as described previously, the calculator waits for the "Device Request" (CFI) which loads the data into the I/O register and clears the "Device Ready" (CEO). The calculator checks the state of "Device Ready" and when it goes false (CEO = HIGH), the calculator knows data is present in the I/O register and proceeds to shift it into the A or B accumulators for processing. The input waveforms are shown in Figure 61.

When blocks of data are to be transferred between a peripheral and the calculator, the interrupt is turned off, and transfer rates as high as 100,000 bits/sec may be possible. Before either input or output of a block of data can start, it is necessary for the calculator to check the status of the peripheral to see if it is turned on and ready. The address locations of the I/O register will remain unchanged during the block transfer. A single I/O instruction shifts the 8 bit byte of data from the 8 least significant locations in A or B to the 8 data locations in the I/O register; gives Device Ready (CEO goes low) 120 nanoseconds after the shift is completed; and shifts the 8 most significant bits in A or B to the 8 least significant locations in A or B in preparation for the next transfer. (Note the address and status field in the I/O register are not disturbed in the shifting). "Device Ready" stays true (low) until the peripheral has received the data and is ready for more. The I/O interface card then returns "Device Request" (CFI) to the calculator. The receiving the "Device Request" (CFI) to the calculator causes loading of the parallel input party lines into the input status and input data locations of the I/O register, and clears the "Device Ready" signal (CEO goes high). The logic sense of "Device Ready" is observed by the calculator and when it goes false (CEO = High) the CPU proceeds to output the next 8-bit byte of data.

If the output I/O interface card is not returning information on the input lines all input lines will be high when the loading, described in the preceding paragraph, takes place. Therefore, if at the beginning the code in the output status field is being used by the I/O interface card and must remain something other than all high it will be necessary for the I/O interface card to receive the output status from the calculator and return it back to the status inputs so that when "Device Request" occurs the status field does not get changed in the I/O register.

Input: After determining if the peripheral is ready to start transferring a block of data the calculator turns off the interrupt and shifts the address code into the I/O Register. (The address code remains unchanged during the block transfer). The "Device Ready" is given (CEO = Low) to the calculator when the 8-bit data byte is ready for input. The "Device Request" signal causes the input data and status to be loaded into the I/O register and causes "Device Ready" to go false (CEO = High). The calculator by checking when "Device Ready" goes false knows the data has been loaded. A single I/O instruction shifts the 8-bit data byte from the I/O register into the 8 most significant locations in the A or B accumulators (Shifting the previous information in A or B 8 places to the right) and causes "Device Ready" to go true (CEO = Low) 120 ns after the last bit has been shifted into A or B. As before if output status is to be retained on the I/O interface card it must be returned to the I/O register upon each input data transfer. Wave forms illustrating high speed operations are shown in Figures 56 and 57.

The calculator software makes use of the interrupt system in two different manners. The first is for remote keyboard like peripherals.

These are those peripherals which logically resemble the calculator keyboard. Only 7-bit ASCII assigned keycodes are recognized by the calculator. The

interrupt takes place by the peripheral indicating to the I/O interface card that a request for service exists. Input data must be stored along with the request for service on the I/O interface card or in the peripheral itself. The stored signal for service is gated with the "Prevent Interrupt" signal through an open collector NAND gate onto the "Service Request" party line ($\overline{SSI} = \text{Low}$ for service). The giving of "Service Request" causes the I/O register to be loaded. However, input data from the I/O interface card is not enabled yet. Thus all status and data inputs are high. This indicates to the CPU that a keyboard-like peripheral is interrupting and address code 12 is shifted into the I/O register. The decoded address 12 on the I/O interface card enables the gates to the data in lines and data is now loaded into the I/O register. After the data has been taken from the I/O register address 12 is again put into the I/O register and "Device Ready" is given as a 360 nanosecond pulse to clear all stored keyboard-like requests for service. This implies all keyboard-like peripherals must be user controlled such that only one interrupt at a time is taking place.

The second is nonkeyboard-like peripherals.

These peripherals will output or enter standard ASCII codes for data by using a special ROM (others ROMs may be developed to handle different codes). When a request for service is given to the I/O interface card by a peripheral the request and all data must be stored until serviced by the calculator. The interface card may have any of 9 addresses (1 to 9). The stored request for service is gated with "Prevent Interrupt" through an open collector NAND gate onto the "Service Request" party line. At the time "Service Request" is recognized address "0" is gated with the stored request for service through an open collector onto an input data or status line which corresponds with the address of the I/O interface card. For example, "Data In" 0 which is the 1st position in the I/O register represents card address 1, and 2nd position is card address 2, etc. When the I/O register is loaded as a result of the "Service Request" the interrupting I/O card's address is loaded into the I/O register and "Prevent Interrupt" enabled ($\overline{SIH} = \text{Low}$). The contents of the I/O register are processed by the CPU which then shifts the interrupting card's address into the I/O register. The address enables the gates to the data-in lines and data is loaded into the I/O register. After the data is processed by the CPU the interrupting card's address is shifted from the CPU into the I/O register and a 360 nanosecond "Device Ready" pulse ($\overline{CEO} = \text{Low}$) given to clear the stored request for service on the I/O interface card, after which the "Prevent Interrupt" is disabled and the next interrupt allowed to take place. Under this system, multiple interrupts may take place without consequence. Each will be serviced in turn from low to high address position. An interrupting peripheral may also interrupt to request output data from the I/O register. The interrupting process is the same as above except the calculator transmits data rather than receives data. Figure 58 shows waveforms illustrating the interrupt.

The following table lists the general I/O instruction set and the associated codes:

632

I/O INSTRUCTION SET

| NAME | INSTRUCTION
EXECUTION TIME | INSTRUCTION CODE | | | | | | | | | | | | | | | | SELECT | CD |
|------|-------------------------------|------------------|----|----|----|-----|----|-----|---|---|---|---|---|---|---|---|---|--------|----|
| | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| STF | 9 μ s | H | H | H | H | - | H | L | H | H | H | H | H | H | H | H | | | |
| CLF | 9 μ s | H | H | H | H | - | H | H | H | H | H | H | H | H | H | H | | | |
| SFC | 9 μ s | H | H | H | H | - | H | H/C | H | H | H | L | | | | | | | |
| SFS | 9 μ s | H | H | H | H | - | H | H/C | H | L | H | L | | | | | | | |
| CLC | 9 μ s | H | H | H | H | - | H | H/C | H | L | H | H | | | | | | | |
| STC | 9 μ s | H | H | H | H | - | H | H/C | H | H | L | L | | | | | | | |
| OT* | 15 μ s | H | H | H | H | A/B | H | H/C | L | L | H | H | | | | | | | |
| LI* | 15 μ s | H | H | H | H | A/B | H | H/C | L | H | L | H | | | | | | | |
| MI* | 15 μ s | H | H | H | H | A/B | H | H/C | L | L | L | H | | | | | | | |

The following describes the function of each I/O instruction with the 5 allowable select codes.

| | | | |
|----|----------------|--|----|
| 5 | STF <SC> | Set the flag. STF is a 240 nanosecond positive true pulse which accomplishes the following with the various select codes. | 5 |
| | STF 00 | Not used by the calculator. | |
| 10 | STF 01 | a. Sets the "Service Inhibit" flip-flop to the true state (SIH=Low; interrupt not allowed).
b. Causes parallel input data and status to be loaded into the I/O register. | 10 |
| | STF 02 | Generates a 240 nanosecond positive true MCR pulse. | |
| | STF 04, 08, 16 | Not used by the calculator. | |
| | CLF <SC> | Clear the flag. CLF is a 240 ns positive true pulse which accomplishes the following with the various select codes. | |
| 15 | CLF 00 | Not used by the calculator | 15 |
| | CLF 01 | a. Clears the "Service Inhibit" flip-flop to the false state. (SIH=High; interrupt allowed).
b. Loads address locations in I/O register gister with 0's. (0=Low)
c. Clears "Device Ready" flip-flop ($\overline{CEO}$ =High). | |
| 20 | CLF 02 | Clears MCR flag flip-flop. | 20 |
| | CLF 04 | Clears PEN flip-flop (PEN=Low). | |
| | CLF 08 | Clears DEN flip-flop ($\overline{DEN}$ =High). | |
| | CLF 16 | Generates a 240 nanosecond positive true KLS pulse. | |
| 25 | SFC <SC> H/C | Skip if flag clear. SFC is a 240 ns positive true pulse which accomplishes the following with the various select codes. If C is given a 240 nanosecond CLF pulse is given after SFC. | 25 |
| 30 | SFC 00 | Causes the next instruction to be skipped if the STOP key has not been depressed. | 30 |
| | SFC 01 | Causes the next instruction to be skipped if Device Ready is true ($\overline{CEO}$ =Low). | |
| | SFC 02 | Causes the next instruction to be skipped if the MCR flag flip-flop is clear. | |
| 35 | SFC 04 | Causes the next instruction to be skipped if the PEN flip-flop is clear. (PEN=Low). | 35 |
| | SFS <SC> H/C | Skip is flag set. SFS is a 240 nanosecond positive true pulse which accomplishes the following with the various codes. If C is given then a 240 nanosecond CLF Pulse is issued after SFS. | |
| 40 | SFS 00 | Causes the next instruction to be skipped if the STOP key is depressed. | 40 |
| | SFS 01 | Causes the next instruction to be skipped if "Device Ready" is false ($\overline{CEO}$ =High). | |

634

622

1,444,141

622

| | | | |
|----|----------------|--|----|
| | SFS 02 | Causes the next instruction to be skipped if the MCR flag flip-flop is set. | |
| | SFS 04 | Causes the next instruction to be skipped if the PEN flip-flop is set (PEN = High). | |
| 5 | CLC <SC> H/C | Clear Control. CLC is a 240 nanosecond negative true pulse and is not used by the calculator. If C is given then a 240 nanosecond positive true CLF pulse is given after CLC. | 5 |
| 10 | STC <SC> H/C | Set the Control. STC is a 240 nanosecond positive true pulse which accomplishes the following with the various select codes. If C is given a 240 nanosecond CLF pulse is issued after STC. | 10 |
| | STC 00 | Not used by the calculator. | |
| | STC 01 | Sets the "Device Ready" flip-flop ($\overline{CEO}$ =Low). | |
| 15 | STC 02 | Generates a 240 nanosecond positive true MLS pulse for the magnetic card reader. | 15 |
| | STC 04, 06, 16 | Not used by the calculator. | |
| 20 | OTX <SC> H/C | Output A or B causes data bits from A or B to be shifted to the I/O register and accomplishes the following with the various select codes. If C is given, a 240 nanosecond CLF pulse is given after OTX is executed. | 20 |
| 25 | OTX 00 | The 8 least significant bits in the A or B register are shifted non-inverted to the 8 least significant locations in the I/O register, and 120 nanosecond after the 8th shift the "Device Ready" flip-flop is set ($\overline{CEO}$ =Low). The 8 most significant bits are shifted right 8 places and the 8 significant bits are recirculated to the 8 most significant locations in the A or B registers. The 8 most significant bits in the I/O register are untouched. | 25 |
| 30 | OTX 01 | Sixteen bits from the A or B register are shifted non-inverted to the I/O register. The data in A or B recirculates. | 30 |
| | OTX 02 | Not used by the calculator. | |
| 35 | OTX 04 | Same as OTX 01 and in addition, 120 ns after the 16th bit has been shifted nanoseconds printer enable flip-flop is set. | 35 |
| | OTX 08 | Same as OTX 01 and in addition, 120 nanoseconds after the 16th bit has been shifted the display enable flip-flop is set. | |
| 40 | OTX 16 | Same as OTX 01 and in addition, 120 nanoseconds after the 16th bit has been shifted the 240 nanosecond signal is generated. | 40 |
| 45 | LIX <SC> H/C | Load into A or B. Loads data bits from the I/O register into the A or B register and accomplishes the following with the various select codes. If C is given, a 240 nanosecond CLF pulse is given after LIX is executed. | 45 |
| | LIX 00 | The eight least significant bits in the I/O register are shifted inverted to the eight most significant locations of A or B, and 120 nanoseconds after the 8th shift the | |

"Device Ready" flip-flop is set (CEO=Low). A or B is shifted right eight places as the I/O register data comes in. The 8 most significant bits in the I/O register are untouched.

| | | | |
|----|--------------------|--|----|
| 5 | LIX 01 | The 16 bits of the I/O register are transferred inverted to the A or B register. Data in the I/O register is lost. | 5 |
| | LIX 02, 04, 08, 16 | Not used by the calculator. | |
| 10 | MIX <SC> H/C | Merge into A or B. Merges data from the I/O register into A or B registers and accomplishes the following with various select codes. If C is given, a 240 nanosecond CLF pulse is given after MIX is executed. | 10 |
| 15 | MIX 00 | The eight least significant bits in the I/O register are merged with the eight least significant bits of the A or B register and shifted to the 8 most significant locations of A or B; 120 nanosecond the merge takes place the Device Ready flip-flop is set (CEO=Low). A or B shifts right 8 places as the data is merged and shifted to the most significant locations. The 8 most significant bits of the I/O register are untouched. | 15 |
| 20 | MIX 01 | The 16 bits of the I/O register are merged with the 16 bits of the A or B register and contained in the A or B register. | 20 |
| | MIX 02, 04, 08, 16 | Not used by the calculator. | |

Examples of various drivers which transfer data are given below:

25 Example 1. 25

Typical Subroutine to Get Status of I/O Device.

Calling Sequence:

| | | | | | |
|----|------|-----|-------------|--|----|
| | | LDB | Select Code | | |
| | | JSM | Stat | | |
| 30 | Stat | STF | 1 | Turn off the interrupt system. | 30 |
| | | OBT | 1 | Load I/O register with select code. | |
| | | STF | 1 | Load I/O register with status of I/O device. | |
| | | LIA | 1 | Load A-Register with status information. | |
| | | CLF | 1 | Turn on interrupt. | |
| 35 | | RET | | Return. | 35 |

Example 2.

Typical Subroutine to Output an 8 bit character.

Calling Sequence:

| | | | | |
|----|-----|---------|-------------------------------------|----|
| | OTA | 1 | Output 16 bits to the I/O register. | |
| 40 | | STC 1 | | 40 |
| | | SFS 1 | Loop until I/O flag is set by the | |
| | | JMP *-1 | output device. | |
| | | CLF 1 | | |



624

1,444,141

624

Example 3.

High Speed Output Where the Calculator is Faster than Output Device.

Calling Sequence:

| | | | | |
|------|-----|------|---|---|
| | ST* | I | —(Number of 16 bit words to be output) + 1. | |
| 5 | ST* | J | Address of first word in the array. | 5 |
| | LDB | SC | Select Code | |
| | JSM | OUT2 | | |
| OUT2 | JSM | STAT | Get status of output device | |
| | RAR | 9 | and position it. | |

10

Example 4A.

10

Typical Subroutine to Input an 8-bit Character.

Calling sequence is:

| | | | |
|----|-----|---|---|
| | LDB | Select code | |
| | JSM | In | |
| 15 | ... | Return is made with the data in the A Register. | 15 |
| In | STF | 1 | Turn off interrupt system |
| | OTB | 1 | Load I/O register with the select code |
| | STC | 1, C | Pulse the flag and turn interrupt system on |
| | JSM | STAT | Get status off the input device |
| 20 | RAR | 9 | and position it. |
| | SAP | *-2, C | If device is busy then continue to loop |
| | SAR | 7 | else position data bits |
| | RET | | Return. |
| | SAP | OUT2 | If device is busy, continue to loop |
| 25 | STF | 1 | Turn off interrupt system. |
| | OTB | 1 | Output select code |
| | LDB | 1 | B←Counter for number of words to be output |
| | LDA | J,I | Load next data word |
| | SEC | *+1, C | E←0 |
| 30 | OTA | 0 | Output 8 bits from A |
| | SFS | 1 | Loop until device sets |
| | JMP | *-1 | flag. |

Example 4A (continued).

| | | | | |
|---|-----|--------|---|---|
| | SEC | *-3, S | If $E=0$ and $E \leftarrow J$ then loop to output last 8 bits | |
| | ISZ | J | Increment array address pointer | |
| | RIB | *-7 | Increment count and loop if not finished. | |
| 5 | CLF | 1 | Turn on interrupt system | |
| | RET | | Return | 5 |

Example 3B.

If the Output Device is Faster than the Calculator then Fewer Instructions can be Used.

| | | | | |
|----|-----|---|-----------------------|----|
| | OTA | 0 | Output first 8 bits. | |
| 10 | OTA | 0 | Output second 8 bits. | 10 |

Example 5A.

High speed input where the calculator is faster than the input device.

Calling sequence:

| | | | | |
|------|-----|--------|--|----|
| | ST* | I | —(Number of 16 bit words to be input) + 1 | |
| | ST* | J | Address | |
| | LDB | SC | Select code | |
| 20 | JSM | In2 | | 20 |
| In 2 | JSM | STAT | Get status of input device | |
| | RAR | 9 | and position it. | |
| | SAP | In2 | If device is busy, continue to loop | |
| | STF | 1 | Turn off interrupt system | |
| 25 | OTB | 1 | Output select code | 25 |
| | STC | 1 | Command device to read | |
| | LDB | I | $R \leftarrow$ Counter for number of words to be input | |
| | SEC | *+1, C | $E \leftarrow 0$ | |
| | SFS | 1 | Loop until input | |
| 30 | JMP | *-1 | device sets flag | 30 |
| | LIA | 0 | Load 8 bits from I/O register | |
| | SEC | *-3, S | If $E=0$ and $E \leftarrow 1$ then loop to input last 8 bits | |
| | STA | J, I | Save data word in array | |
| | ISZ | J | Increment array address pointer | |
| 35 | RIB | *-7 | Increment count and loop if not finished. | 35 |

Example 5A (continued).

CLF 1 Turn on interrupt system
RET Return

Example 5B.

5 *If the input device is faster than the calculator then the number of instructions can be reduced.* 5

LIA 0 Input first 8 bits
LIA 0 Input second 8 bits

10 All output I/O interface cards which are to be fully interchangeable with both the present and other calculators must have storage either on the I/O interface card or in the peripheral to which information is being transmitted. 10

15 Blocks (A) and (B) are the storage latches which store information coming from the I/O register. When the output of gate (C) goes high, data is latched; when low, the outputs of the latch track the inputs. Gates (D) decode the address code (14=1110) and pass it positive true to gate (E). "Device Ready" (CEO) is also passed positive true to gate (E). Gates (H) are open collector and pass status and "Device Request" (CFI) onto the input party lines. 15

20 An example of a calculator output would be: Output the address 14 which enables status gates (H) and see if the power is on. If on, output address, status, and data to gates (A), (B), and (D). The output of (C) is low allowing data and status to pass. Next give "Device Ready" (CEO=Low); this enables flip-flop (G), clocks flip-flop (F) which causes (A) and (B) to latch, and sends control to the peripheral. The peripheral acknowledges receipt of control by returning FLAG (FLAG=High) in a busy state this continues to keep (A) and (B) latched and clears control flip-flop (F). When the peripheral is done acting, the FLAG is returned to the not busy state (FLAG=Low) which clocks flip-flop (G) and causes output at (C) to go low enabling (A) and (B). The output of (G) drives the CFI gate which has been enabled from (E) and CFI goes low. CFI is received by the calculator which responds by returning CEO high. This causes the output of (E) to go low, clearing flip-flop (G) and returning CFI high. This completes 1 output cycle. 20

25 All input I/O interface cards which are to be fully interchangeable with both the present and other calculators must have storage either on the I/O interface card or in the peripheral from which information is being received. 25

30 Block (A) is used to store information coming from the peripheral. (B) stores status coming from the I/O register which may be needed by the peripheral. The output tracks the input whenever the enable on the latch is low. Block (C) decodes the address code into one of 10 addresses which are jumper selectable. An example of a calculator input would be as follows: the address code would be decoded by (C); the calculator would load status through the open collector input status gates (D). If the peripheral is on and ready, the address code and output status (if necessary) would be sent to (B) and (C). The decoded address is passed, positive true, to gate (E). The enable at (B) is low so that status is passed to the peripheral. The "Device Ready" is given (CEO=Low) and comes to (E) positive true. The output of (E) clocks flip-flop (F) through gate (H). The output of (F) gives control to the peripheral and also enables (A) to receive data. The peripheral responds in a busy state (FLAG=High). When data is ready to be input the FLAG is driven low. Data is latched when the FLAG goes low in (A). Also when FLAG goes low, (G), having been enabled by the output of (H), is clocked driving (J) from its Q output. (I) is enabled by the output of (H) and so CFI is driven low. Data is loaded into the I/O register from open collector gates (I) and CEO driven high as a result of the calculator receiving CFI. This clears flip-flop (G) and disables the input gates (I) completing an input cycle. 30

35 Figure 59 illustrates the logic required on an I/O interface module to interface the calculator with an external X-Y plotter. 35

40 Figure 60 illustrates the logic required on an I/O interface module to interface the calculator with an external line printing unit. 40

45 Figure 61 illustrates the logic required on an I/O interface module to interface the calculator with an external modem for transmitting and receiving information via telephone lines. 45

50 Figure 62 illustrates the logic required on an I/O interface module to input or output any eight-bit code. 50

A power preset circuit is employed on interface modules using the interrupt

system to prevent an interrupt when the peripheral power is turned off or on. This can usually be done by sensing the peripherals' +5 volts and presetting when the voltage drops below 3 to 4 volts.

An example of a calculator interrupt would be as follows: (B) may be clocked at any time storing the data is (E) and (F). The calculator enables the interrupt to take place by making "Prevent Interrupt" false (SIH=High) and outputting address 0 to decoder (L). (G) is enabled when SIH goes high through gate (M) causing SSI to be driven low. The calculator responds by loading the I/O register. Gates (H) are inhibited by gates (N) and (J) and gate (K) is enabled because of address 0, thus DI0 is the only true signal loaded into the I/O register. The calculator interprets this to mean the I/O interface card at address 1 has caused the interrupt. The calculator outputs address 1 to the decoder which enables gates (H) with (N) and (J) and then loads the data. After the data is stored the calculator outputs address 1 and sends "Device Ready" (CEO=Low) as a 360 nanosecond pulse which is used to clear (B) through gates (O) and (D). This completes an input cycle.

KEYBOARD INPUT UNIT

The keyboard input unit is shown in Figure 63 and is designed around a 128-position matrix. Each position is scanned sequentially. When a key is depressed, the counter, which drives the scanner, is stopped. The address of the 7-bit counter corresponds to the ASCII code of the key depressed. The entire scan period is four milliseconds. Two key rollover is incorporated in this design.

When a key is depressed, SSI is generated unless inhibited by either SIH or KLS. As soon as the service request is acknowledged, the CPU will give a select code of 12 which will gate the ASCII keycode onto the data lines.

Another feature of this keyboard is automation repeat. If a key is depressed for more than 1.5 seconds, the keycode will be entered repeatedly at a rate of 15 entries per second until the key is released.

OUTPUT DISPLAY UNIT

Referring to Figures 72A—F and 73A—B, there is shown the hardware associated with the calculator display. The display comprises a single register 400 of thirty-two alpha-numeric characters, each character position of which is a seven row by five column matrix of light emitting diodes (LED). In addition to the display hardware illustrated, the complete calculator display system comprises a firmware display routine which is detailed on page 35 of the basic system firmware listing and an I/O register shown in Figures 49A—D. The firmware display routine delivers a sixteen-bit word to the display circuitry. Four of these bits 402 and 404 are decoded into one of sixteen character positions. The remaining twelve bits 422 are two six-bit ASCII coded characters which are alternately decoded in ROM 430 into their row and column information. A seven-bit data latch 432 is provided to allow the ROM 430 to decode both the nth and the (n+16)th characters which are displayed simultaneously.

An asymmetrical clock circuit 426 drives input gating circuitry 436 and counter-decoder circuitry 434 to decode the row information of the nth and the (n+16)th characters. Protection circuitry 428 is provided to blank the display in case of failure of either logic signal DEN 424 or CA0 438.

Three-bit column data 410 and three of the four-bit character position bits 402 are applied to a one-of-four decoder circuit 420. The output of circuit 420 is fed to the column drivers, 418. Four columns eight emitting diodes 400 are driven by each of the forty column driver. Thus, all, one hundred sixty columns of the display can be scanned. The remaining character position but 404 enables alternate sections of row drivers 416 to complete the one-of-sixteen character position decoding.

MAGNETIC TAPE CASSETTE READING AND RECORDING UNIT

The magnetic tape cassette reading and recording unit is shown in the block diagram of Figure 64 and in the detailed schematic diagram of Figures 65—69. Operation of the magnetic tape cassette reading and recording unit is largely automatic. It is only necessary to specify the type of operation to be performed and the limits desired. The commands for doing this may either be entered directly from the keyboard input unit or as part of a program. The calculator then determines the necessary commands required to cause the magnetic tape cassette reading and recording unit to perform the desired operation.

Several modes of operation are possible. Secure and unsecure programs, data, and sets of user definable key definitions may be recorded on the magnetic tape and subsequently loaded back into the calculator.

Referring now to Figures 64 and 65, the operation of the interface portion of the magnetic tape cassette reading and recording unit will be described.

I. Address Decoder

IC 6 and IC 7 are used to detect a select code of 10 when the cassette is accessed. When a select code of 10 is detected, and the control line CEO is true the cassette is enabled.

II. Enabling Input Data Bits

The open collector nands in IC 2 and IC 3 allow the input data bits (ID 0 through ID 7) to be gated onto the calculator input bus when the card is selected by the address and control line as described in Section I.

III. Enabling Input Status Bits

The open collector nands in IC 1 allow the input status bits (IS 0 through IS 3) to be gated onto the calculator input bus when a select code of 10 is given and the control line is true.

IV. Flag

D type flip-flop 10 is the flag flip-flop. It is held cleared except when the I/O card is selected as described in Section I. When the I/O card is selected the flip-flop is set by the trailing edge of the flag pulse from the cassette. In addition, the flag flip-flop is held preset (at pin 5) when the cassette door is open, clear leader is detected, or an interrupting character is read. A simultaneous preset and clear results in a true output from the flip-flop. The flag flip-flop signal is gated to the calculator (CIF, pin 2-m) by the open collector nand gate only when the I/O card is selected.

V. Interrupt

D-type flip-flop 13 is the interrupt flip-flop. It is held cleared whenever the I/O card is selected as described in Section I. The interrupt is allowed only if the cassette is in control mode (pin 2—3). An interrupt signal comes whenever the cassette goes onto clear leader, the cassette door is opened, or an interrupting character is read. An interrupting character is a character with data bits 2, 3, 4, 5 all ones, read when the cassette is in control mode. When the signal Service Interrupt Inhibit is false (SIH=1, pin 2—1) the interrupt signal is gated through IC 4 pin 6 and 8 to the calculator. The data bit SI 1, (Pin 1—4) is pulled low to tell the calculator the address of the peripheral which interrupted.

Referring to Figures 64, 66A—B, and 66' the operation of the control logic portion of the magnetic tape cassette reading and recording unit will be described.

1.1 Power on Preset

Transistor Q1 and diodes CR1 and CR2 are part of the power on circuit which makes sure that the cassette powers up in the proper modes.

1.2 Control Signal

The control signal (YCNT on pin 11) comes from the Interface card as a positive true signal telling the Logic card to process the command on the input lines. The positive going edge of the control signal fires the one-shot (IC 7) giving a 300 nsec. control Pulse used to strobe storage elements.

1.3 Decode and Storage of Commands

The 1-of-10 open collector decoder (IC 19) is strobed on its D input by the control pulse. (See Section 1.2) On the A, B, and C inputs of the decoder are output status bits OS0 through OS2. Those output status bits contain commands for the transport as follows:

| | OS2
(Write) | OS1
(Reverse) | OS0
(Fast) | Command | |
|----|----------------|------------------|---------------|--------------------------|----|
| | 0 | 0 | 0 | Read-Forward-Slow | |
| | 0 | 0 | 1 | Read-Forward-Fast | |
| 5 | 0 | 1 | 0 | Read-Reverse-Slow | 5 |
| | 0 | 1 | 1 | Read-Reverse-Fast | |
| | 1 | 0 | 0 | Write-Forward-Slow | |
| | 1 | 0 | 1 | STOP | |
| | 1 | 1 | 0 | Continue Present Command | |
| 10 | 1 | 1 | 1 | Continue Present Command | 10 |

Output status bit 3 (OS3) indicates if the cassette should be in control mode or data mode. Control mode is explained in Section 1.12.

If when the decoder is strobed, the command on the output status lines is other than "STOP" or "CONTINUE", the open collector wired-or decoder outputs 0 through 4 are low for the width of the control pulse. This wire-or output pulse strobes the command on the output status bits into the Quad Latch, IC20, and preset the run flip-flop 15—2. The command on the run lines are gated off the logic card to the motor control card, and cassette motion begins.

If the command is "STOP", the decoder output 5 (pin 6) goes low, which clears both the run flip-flop (15—2) and the Quad Latch (IC20). If the command is "CONTINUE", both the Quad Latch and the run flip-flop remain unchanged.

The run flip-flop and the Quad Latch are also cleared by power on preset.

1.4 Run Flip-Flop

The run flip-flop (1502) is preset by the issuing of a command as described in Section 6.3. It is cleared by an interrupt signal from the I/O card, a stop command, power on preset, or a cassette not being in place. The run flip-flop is also cleared by the clear leader signal on its clock input, whenever the cassette runs into clear leader. Since the clock input of the flip-flop is edge sensitive, the transport is stopped only when it first goes onto clear leader, and tape motion is allowed by the issuing of a new command even when on clear leader.

The Schmidt-triggered gates 1—1 and 1—2 are used to filter noise on the interrupt, clear leader, and cassette in place lines.

1.5 Rewind Mode

D type flip-flop 15—1 is the rewind flip-flop. It is held preset as long as the rewind button is held in. It is cleared by clear leader detection, cassette not in place, power on preset, stop command, or on interrupt from the I/O card. The rewind flip-flop is also cleared through its clock input, whenever a new command is issued. A simultaneous preset and clear results in the Q output of the flip-flop being true. When the transport is stopped (flip-flop 15—2 cleared) and the rewind flip-flop is set the output of gate 16—3 will be true, which will force the NOR gates 17—1, 17—2, and 17—3 to 0 outputs. These outputs will cause the tape to be rewound in high speed reverse to clear reader, unless the rewind flip-flop is cleared as described above.

1.6 Transport Status

Input status bits; YIS3 (pin F), YIS2 (pin 6), YIS1 (pin E), and YIS0 (pin 5), reflect the status of the transport as follows:

YIS3 — Negative true, cassette in place
 YIS2 — Positive true, clear leader detected
 YIS1 — Negative true, writing on cassette permitted
 YIS0 — Positive true, control mode

1.7 Internal Clock

The internal clock used for writing data is generated by the two one-shots IC6 and IC13. The clock is not symmetrical, but IC6 is true for 133 μ sec and IC13 is true for 220 μ sec. Padding resistor R10 is then used to adjust the total time of 333 μ sec for a 3KC clock rate. The internal clock is enabled only when the cassette command is write as detected at pin 5 of IC13.

1.8 Nine Bit Shift Register

A nine bit shift register is made from IC3, IC4, and flip-flop 5—2. In write operations this register is loaded parallel by a pulse from gate 16—1, and then shifts the data serially to IC3 pin 10 for writing on the tape. In read mode the data is clocked from the tape, serially, to IC5 pin 12 where it is serially loaded into the register. From the shift register it is read as parallel data, at the flag, by the calculator.

1.9 Data, Clock Mark Sequence

During write operation a write clock, a write data, and a write mark signal are sent from the logic card. During read operations a read clock, a read data, and a read mark signal are received by the logic card. The data lines are NRZ; clocked by the appropriate clock. When clock pulses and marker pulses are properly sequenced they constitute a character. The sequence is 9 data bits per character, separated by marks.

Mark - 9 data bits - mark - 9 data bits - mark.

BIT-MARK-SEQUENCE CODE

| Three types of information | Track |
|----------------------------|-------|
| "1" | A |
| | B |
| "0" | A |
| | B |
| Mark | A |
| | B |

10 bit character

M D₇ D₆ D₅ D₄ C D₃ D₂ D₁ D₀ M

8 data bits

1 marker

1 control bit

1.10 Divide by 10 Counter During Write

The divide by 10 counter (IC14) is used to sequence the clock and mark pulses during write (See section 6.9 for a sequence description). When the counter is at counts 0 through 8 the internal clock pulses generated by IC6 and IC13 are gated through gate 18—4 from where it goes off the card as the write clock, and through gate 18—2 to shift the shift register. When the divide by 10 counter is at count 9, the internal clock pulse is inhibited at gate 18—4, but is enabled at gate 10—2 from where it passes off the board as a write mark pulse. Since the counter "wraps around" from count 9 to count 0, we get the repeating sequence, marker — 9 bits — marker.

Each time the counter goes to 9, a character is completed and the write mark pulse, in addition to going off the board, is gated through gate 12—3 to make up a flag, telling the calculator that the character has been written. The calculator will then send another parallel character to the shift register and the sequence will be repeated.

Special conditions prevail for the writing of the first of a string of characters. When the transport goes from read to write mode, flip-flops 8—1 and 8—2 are set in their edge sensitive clock inputs. Flip-flop 8—2 presets the counter to count 9, making sure the sequence begins with a mark pulse. The same flip-flop 8—2 is cleared by this first mark pulse, after which the counter continues its "wrap around" sequence. Flip-flop 8—2 being set, inhibits this first leading write mark pulse from passing through gate 12—3 and becoming a flag. Flip-flop 8—2 is cleared by the first write pulse, after which the write sequence continues as normal.

I.11 Divide by Ten Counter During Read

Only a sequence of 9 data bits followed by a mark is to be recognized as a character during read. The divide by 10 counter is used to detect this sequence.

Whenever the transport is stopped, flip-flop 5—1 is held preset. The output of flip-flop 5—1, presets the counter to a count of 0. Flip-flop 5—1 is cleared with the leading edge of the read date clock and the counter counts on the trailing edge. The read date pulses are counted by the counter and also pass through gate 18—2 to shift the shift register. When the counter has counted 9 read pulses, its count of 9 puts a true signal on pin 13 gate 19—1. If a read mark pulse comes next, pin 1 of gate 19—1 is true for the width of the read mark pulse. These two signals make up part of the flag signal which tells the calculator a valid character is in the shift register. The trailing edge of the read marker pulse, sets flip-flop 5—1, which presets the counter to 0 and the sequence is ready to be repeated.

If a marker is read when the counter is at a count other than 9; no flag is given, flip-flop 5—1 is set, and the counter is preset to 0. If 10 read pulses come in a row, the counter wraps around to 0 and no flag is given. Thus nothing but 9 read clock pulses, followed by a read mark pulse, generates a flag and is recognized as a character.

I.12 Control Mode and Flags

Each character contains 8 data bits. The middle bit of the 9 bit character is a control bit. If the center bit is a 1, the character is a control character. If the center bit is a 0, the character is a data character. If the transport is in control mode during write, the center bit is loaded as a 1 at IC4 pin 5, and the character is written as a control character. In data mode the center bit of each character written is 0.

If the transport is reading in control mode, flags are sent to the calculator only when a control character is in the shift register. If the transport is in control mode, and a data character is in the shift register, the read flag is inhibited at IC9—1 pin 2. This IC9—1 pin 2 is always true, except when the cassette is in control mode and the center bit of the shift register is 0. In data mode, flags are sent to the calculator for both data and control characters.

The flag circuitry on the interface card is trailing edge sensitive, making sure the character is completely written or completely read before the calculator is flagged.

Referring now to Figures 64 and 67A—B, the operation of the read/write portion of the magnetic tape cassette reading and recording unit will be described.

I. General Description

The read/write board has two main functions. In the WRITE mode it encodes bit serial data into two-channel Bit-Mark-Sequence (BMS) data to be fed to the head driver and written on tape.

In the READ mode it decodes the two-channel analog BMS data from the head preamplifier into clock, mark, and bit-serial data pulses from the tape signals.

II. WRITE Mode

When Write Permit is true (YWPT=1) and Write Command is true (YWTC=1) then Write Enable is true (YWEN=1; Q7 on) and writing on the tape is allowed. The 3-input NAND gates of IC2 are used to encode the bit-serial data (YWDI) into BMS. Logic 1's (YWDI=1) are written on Channel A. Logic 0's (YWDI=0) are written on Channel B. A mark is written when Write Mark is true (YWMK=1). The data and marks are clocked in by means of the Write Clock (YWCL).

III. READ Mode

The two amplifiers of IC6 and their associated circuitry comprise two threshold detectors that convert the analog signals from the head (ARA, ARB) to

digital signals. The amplifiers switch between their positive and negative saturated states. The thresholds are adjusted so that the amplifiers switch states when the analog signal is approximately 30% of its peak value measured near BOT, moving forward.

The head signal amplitude is proportional to tape speed. Therefore, the thresholds are adjusted high for fast tape motion and low for slow tape motion by switching Q1 and Q2 with the Fast Command (NFTC) line. The positive and negative references for the thresholds are derived from the 12 volt power supplies.

The two channels of digital data are gated to the decode circuitry through IC1. The decoder consists of IC3, IC5, IC8, and IC9.

IC8 is a 0.5 μ s one-shot that is fired once for each data bit or mark that is gated in from the threshold detectors. This one-shot pulse comprises a read clock signal that is gated to NRCL by IC5 when the bit associated with the pulse is not a mark.

IC9—1 is a flip-flop that serves three functions. First, it provides a Read Mark pulse (YRMK) whenever a mark is read from the tape. Second, it controls IC5 to allow the one-shot pulse to appear as Read Clock (NRCL). Third, it controls IC3 and IC5 to allow data to be gated through to the data output flip-flop, IC9—2.

Referring now to Figures 64 and 68A—B, the operation of the motor control portion of the magnetic tape cassette reading and recording unit will be described.

I'. Speed Reference

Resistors R1, R2, and R3 are used to generate a reference voltage proportional to the desired motor speed. The voltage seen at the positive input of U3 is about 3.0 volts for low speed and 8.0 volts for high speed.

II'. Comparator-Amplifier

Op Amp U3 compares the speed reference with a signal representing the actual motor speed (see Item IV' below). The difference signal is amplified and fed to the driver circuitry which increases or decreases the drive voltage available to the motor.

Resistor R8 and capacitor C1 create negative feed back around U1, resulting in a DC gain of 26 db and a single pole at about 100 Hz. This "tailors" the servo loop frequency response to provide stable operation with rapid error correction.

III'. Motor Driver

Diode D1 shifts the DC drive level by about 7 volts. Transistors Q1, Q2, and the Motor Pass Transistor (located on the Regulator Board) furnish current gain to drive the motor. Resistor R7 ensures turnoff for the Motor Pass Transistor. Transistor Q3 is used to dynamically brake the motor if its speed is greater than desired.

Diode D3 reduces the maximum voltage at the motor when on clear leader. This voltage limit effectively reduces the motor torque, preventing damage to the motor, friction drive, or tape cassette should the tape be pulled against the hub. Diode D2 prevents damage to Q1 during this reduced torque condition.

IV. Back-EMF Amplifier

The motor terminal voltage is composed of two parts — the "IR" voltage drop in the motor armature resistance, and the motor generated voltage, or Back-EMF. Back-EMF is directly proportional to motor speed, and is used as the motor speed feedback signal.

Sense resistor R9 produces a voltage proportional to the motor armature current, and therefore proportional to the "IR" voltage drop. Op amp U4 and resistors R10 thru R14 subtract the "IR" voltage drop from the motor terminal voltage, resulting in a voltage proportional to the motor speed. This signal is about 3.0 volts for low speed and 8.0 volts for high speed; it is fed back and compared with the reference voltage, as described in Item II' above.

The process of sensing the "IR" voltage drop, as described above, is accurate only for a single winding temperature. Because lower armature resistances (caused by lower temperatures) could cause instability in the servo, this "perfect compensation" point is placed at the bottom of the operating range — 0°C in this case. Increasing temperature causes reduced load regulation (greater motor current causes reduced speed), but servo operation remains stable.

V'. Motor and Solenoid Selector

The selector circuitry activates the proper motor and solenoid. When in "Run

Fwd" mode, Q4, Q5, and Q8 are saturated "on"; in "Run Rev" mode Q6, Q7, and Q9 are saturated. When in "Stop" mode, all devices are off.

Referring now to Figures 64 and 69, the operation of the interconnect portion of the magnetic tape cassette reading and recording unit will be described.

I. Connection From Transport to Mother Board

The main function of the Interconnect Board is to electrically connect the transport mechanism to the Mother Board. There are four groups of wires coming from the transport: the Motor wires, the Solenoid and Switch wires, the Head Board wires, and the Photosensor wires. Each group of wires is terminated on a Pin Board which plugs into the Interconnect Board; this partitioning allows for ease of assembly and service.

II. Clear Leader Signal

Most of the circuitry on the Interconnect Board is used to generate the "Clear Leader" signal. The photosensor assembly on the transport contains an incandescent lamp and a photoconductive cell. When magnetic tape is over the photosensor, no optical coupling occurs, resulting in a high photoconductor impedance. When clear leader is over the photosensor, light is reflected off the light colored plastic of the cassette and onto the photoconductor, resulting in a low photoconductor impedance.

Integrated circuit U1 and resistors R4, 5, and 6 form a comparator which switches when the photoconductor impedance is approximately $25k\sim$. Capacitor C5 filters the output of U1 to eliminate false signals of less than 5ms. duration; transistor Q1 amplifies this signal.

Under certain conditions when the tape stops or changes direction a tape loop may form over the photosensor; a reflection can result which causes the same photoconductor impedance as a clear leader. IC's U2, 3, and 4 are used to prevent a "tape loop" from giving a false clear leader indication. Flip-flop U4 stores the clear leader signal; this flip-flop may be turned "on" when the cassette has been running on clear leader for the duration of one-shot U2 (85 ms). This delay period ensures that any loop has been pulled out of the tape, preventing reflection problems.

A "mag tape" (Clear Leader Not) indication is always accurate; therefore it directly resets flip-flop U4. The gates tied to the preset of U4 force the flip-flop "on" during power turn on and when the cassette loader is open; if the tape is, in fact, on "mag tape", this signal persists after preset is removed and resets U4.

III. Solenoid Turnoff

Diodes D1 and D2 and resistor R1 limit the peak solenoid flyback voltage during turnoff to about 24 volts. This also provides rapid solenoid dropout by applying up to -12 volts across the solenoid during turnoff.

IV. Motor Padding Resistors

Resistors R2 and R3 are selected to trim a given motor to between 10.30 and 10.60 ohms effective armature resistance. Capacitors C1 and C2 are used to suppress motor brush noise.

V. Cassette Sense Switches

Resistors R11 and R12 and "pullup" resistors for the mechanical cassette sense switches, "Cassette In" (positive true) and "Write Permit" (positive true) are the signals generated.

POWER SUPPLY

The power supply system employed in the calculator is constructed as shown in Figures 74 and 75A—B. As shown in Figures 75A—B, a centertapped transformer secondary is connected to supply the unregulated DC voltages indicated. Referring to Figure 75A, the AC voltage from the transformer is rectified by diodes 454 and 456 and filtered by capacitor 478. The output of this rectifier/filter circuit is nominally 19 volts DC at 2.7 amps with a 2 volt peak-to-peak ripple. Transistors 440 and 442 serve as a switch to connect the five-volt output bus to the 19 volt unregulated supply through inductors 450 and 452. Diode 444 serves to clamp the input of inductor 450 to ground when transistors 440 and 442 are switched off. Current flow in inductors 450 and 452 is substantially constant and equal to the load current.



Loss in high current transistor 440 is minimized because it can be completely saturated. Loss in driver transistor 442 is minimized because it can also be saturated. Resistor 446 limits the maximum drive current to transistor 440. Losses in resistor 446 can be minimized by proper positioning of the tap on inductor 450 consistent with transistor parameters and circuit requirements.

Integrated circuit 448 is a linear differential amplifier to drive Q1 and Q2. Any differential amplifier with sufficient voltage capability and bandwidth will work. Since the amplifier employed is linear, R7 and R4 have been included in the circuit to provide sufficient hysteresis for reliable switching. This hysteresis stabilizes the switching frequency and thus stabilizes the switching losses.

Because hysteresis has been added to the circuit, a significant ripple signal (at switching frequency) must be present on the feedback signal to the amplifier. This need for a ripple signal limits the amount of capacitance that can appear between the output of inductor 450 and ground. Inductor 452 serves to isolate this point from the rest of the system. The amount of capacitance that can appear between the output of inductor 452 and ground is essentially unlimited and significantly reduces power supply ripple, and greatly improves response to load transients.

The second winding of inductor 452 is a path for the feedback from the remote sensing. The required ripple signal is added to the feedback signal by transformer action in inductor 452.

The power supply also includes an overvoltage crowbar circuit comprising transistor 458, diode 460, and resistor 462 and a short circuit shut-down circuit (using transistor 464). In the event that the +5 volt bus is grounded, or the crowbar is triggered, transistor 464 saturates and locks integrated circuit 448 off.

The resistor 466 makes a current generator of integrated circuit 448. Resistors 468 and 470 discharge the bases of transistors 450 and 452, respectively. Integrated circuit 472 and its associated components generate a "power on pulse", POP, to initialize the instrument. Integrated Circuit 448 is referenced and powered from an external +12 volt supply. Powering the IC from +12 rather than the unregulated +19 reduces power dissipation in IC 448.

The +12 volt supply of Figures 75A—B references the -12, +5, and +16 supplies directly. The +12 amplifier 448 may be biased either from the unregulated supply for the +12 volt supply or from the operating +16 volt supply. Diodes 474 and 476 determine the appropriate source. This provides a greater power supply margin for the +12 volt supply.

All supplies except the +20 volt supply are current limited. All supplies except the +20 volt supply are crowbar protected against over-voltage.

TYPEWRITER INTERFACE

This interface couples the Facit-Odhner model 3841 (Facit is a RTM) output typewriter to the calculator.

The unit mounts directly on the back of the typewriter. Communications with the calculator are made through about five feet of cable which is terminated by the I/O plug containing a board for buffering and some logic.

Referring to Figures 76A—82, characters from the calculator appear on the data lines as ASCII codes. These codes are recoded by a ROM into the six bit Facit typewriter code for the 46 type bars, and one bit for upper case shift. Functions such as space, tab, line feed, etc. are recoded for easy recognition in the interface since each function must be driven by a separate line. A data latch after the ROM holds codes for processing. If new data arrives during this processing, the two codes are compared to determine if they both drive the same type bar and if they are both numbers. Non-repeating numbers can be typed at 14.5 characters per second, otherwise typing speed is 12 characters per second (reduce these speeds 17% for 50HZ operation). Codes in the latch are gated to the program solenoids or the function solenoids by the control logic.

To understand the coding, notice that two blocks of codes on the Facit typewriter code map are empty. If all function codes are put in these blocks, they can be identified by control logic by testing for (6-4). Each function code puts a 1 on one of five lines and this line opens the correct solenoid gate. Bit 8 is used to discriminate between two sets of function gates. In the case of a program solenoid code, bit 8 identifies numerals.

The control clock is provided by a synchronizing pulse which is generated in the typewriter by a vane wheel attached to the end of the main drive shaft. The vanes interrupt a light beam. When a type cycle is initiated, a modulo eight counter counts synchronizing pulses and the count is decoded by a 1-of-8 decoder.

At each of the eight states, combinational logic can enable solenoid gates, set or clear flag flip-flops or change the counter to state zero, or state 6, or inhibit the counter.

The tables below contain a guide for interpretation of bit pattern data as well as the actual bit patterns for ROM #10 and ROM #11 as shown in Figure 81.

INTERPRETATION OF BIT PATTERN DATA (Bipolar ROM of Figure 81)

1. *FORMAT*

The bit pattern information is in the following format:

$X_1X_2X_3 \dots X_5X_6X_7$ B B $X_{10}X_{11}X_{12}X_{13}$ B $X_{15}X_{16}X_{17}X_{18}$ B... $X_{45}X_{46}X_{47}X_{48}$

A. $X_1X_2X_3$ — Three digits indicating the address (decimal) of the first word of that line. *1

B. $X_5X_6X_7$ — Three digits indicating the address of the last word in that line. *1

C. $X_9X_{10}X_{11}X_{12}$ — Four characters indicating the output states of the first word of that line (corresponding to address $X_1X_2X_3$). *2

D. $X_{15}X_{16}X_{17}X_{18}$ through $X_{45}X_{46}X_{47}X_{48}$ indicate successive output states. *2

E. $X_{45}X_{46}X_{47}X_{48}$ — Four characters indicating the output states of the last word of that line (corresponding to address $X_5X_6X_7$). *2

F. B = Blank or space between group of characters.

2. *TRUTH TABLE*

Logic level definition

L — Output Low (Logic 0)

H — Output High or Open Collector (Logic 1)

X — Don't Care — Output may be High or Low

*1. Addresses are the digital equivalent of the binary address $A_7A_6A_5A_4A_3A_2A_1A_0$. Where a low input address equals a binary 0 and a high input equals a binary one.

*2. Groups of output states are listed $O_4O_3O_2O_1$ respectively.

650

BIPOLAR ROM 10 OF FIGURE 81

| | | | | | | | | |
|---------|------|------|------|------|------|------|------|------|
| 000-007 | LLLL | LLLH | LLLL | LEEL | LLLL | LLLL | HLLL | LHLL |
| 008-015 | HLLL | LLLL | LHLL | LLHL | LLLL | LLHL | LLLL | LLLL |
| 016-023 | LLLL | LELL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 024-031 | LLLL | LELL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 032-039 | LLLH | LLLH | LLHL | LLHH | LHLL | LHLH | LHHL | LHHH |
| 040-047 | HLLL | HLLH | HLHL | HLHH | HHLL | HHLH | HHHL | HHHH |
| 048-055 | LELL | LLLH | LLHL | LLHH | LHLL | LHLH | LHHL | LHHH |
| 056-063 | HLLL | HLLH | HLHL | HLHH | HHLL | HHLH | HHHL | HHHH |
| 064-071 | LLLL | LELL | LLHL | LLHH | LHLL | LHLH | LHHL | LHHH |
| 072-079 | HLLL | HLLH | HLHL | HLHH | HHLL | HHLH | HHHL | HHHH |
| 080-087 | LLLL | LLLH | LLHL | LLHH | LHLL | LHLH | LHHL | LHHH |
| 088-095 | HLLL | HLLH | HLHL | HLHH | LLLL | HHLL | HHHL | LLLL |
| 096-103 | LLLL | LLLH | LLHL | LLHH | LHLL | LHLH | LHHL | LHHH |
| 104-111 | HLLL | HLLH | HLHL | HLHH | HHLL | HHLH | HHHL | HHHH |
| 112-119 | LLLL | LELL | LLHL | LLHH | LHLL | LHLH | LHHL | LHHH |
| 120-127 | HLLL | HLLH | HLHL | HLHH | LLLL | HHLL | HHHL | LLLL |
| 128-135 | LLLL | LELL | LLHL | LLHH | LLLL | LLLL | LLLL | LLLL |
| 136-143 | LLLL | LELL | LLHL | LLHH | LLLL | LLLL | LLLL | LLLL |
| 144-151 | LLLL | LELL | LLHL | LLHH | LLLL | LLLL | LLLL | LLLL |
| 152-159 | LLLL | LELL | LLHL | LLHH | LLLL | LLLL | LLLL | LLLL |
| 160-167 | LLLL | LELL | LLHL | LLHH | LLLL | LLLL | LLLL | LLLL |
| 168-175 | LLLL | LELL | LLHL | LLHH | LLLL | LLLL | LLLL | LLLL |
| 176-183 | LLLL | LELL | LLHL | LLHH | LLLL | LLLL | LLLL | LLLL |
| 184-191 | LLLL | LELL | LLHL | LLHH | LLLL | LLLL | LLLL | LLLL |
| 192-199 | LLLL | LELL | LLHL | LLHH | LLLL | LLLL | LLLL | LLLL |
| 200-207 | LLLL | LELL | LLHL | LLHH | LLLL | LLLL | LLLL | LLLL |
| 208-215 | LLLL | LELL | LLHL | LLHH | LLLL | LLLL | LLLL | LLLL |
| 216-223 | LLLL | LELL | LLHL | HLHH | HLHH | LLLL | HHHL | HHHL |
| 224-231 | LLLL | LELL | LLHL | LLHH | LLLL | LLLL | LLLL | LLLL |
| 232-239 | LLLL | LELL | LLHL | LLHH | LLLL | LLLL | LLLL | LLLL |
| 240-247 | LLLL | LELL | LLHL | LLHH | LLLL | LLLL | LLLL | LLLL |
| 248-255 | LLLL | LELL | LLHL | LLHH | LLLL | LLLL | LLLL | LLLL |

BIPOLAR ROM 11 OF FIGURE 81

| | | | | | | | | |
|---------|------|------|------|------|------|------|------|------|
| 000-007 | LLLL | HLLL | LLLL | LLLL | LLLL | LLLL | HLLL | HLLL |
| 008-015 | LLLL | LHLL | LLLL | HLLL | HLLL | LLLL | LLLL | LLLL |
| 016-023 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 024-031 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 032-039 | LLLL | LHLH | LHLH | LHLH | LHLH | LHLH | LHLH | LHLH |
| 040-047 | LHLH | LHLH | LHLH | LHLH | LLH | HLLH | LLH | LLH |
| 048-055 | HLLH | HLLH | HLLH | HLLH | HLLH | HLLH | HLLH | HLLH |
| 056-063 | HLLH | HLLH | HLLH | LLH | LHLH | LHLH | LHLH | LHLH |
| 064-071 | LLHL | LHHL | LHHL | LHHL | LHHL | LHHL | LHHL | LHHL |
| 072-079 | LHHL | LHHL | LHHL | LHHL | LHHL | LHHL | LHHL | LHHL |
| 080-087 | LHHH | LHHH | LHHH | LHHH | LHHH | LHHH | LHHH | LHHH |
| 088-095 | LHHH | LHHH | LHHH | LLHH | LLLL | LLHH | LLHH | LHLH |
| 096-103 | LHHL | LLHL | LLHL | LLHL | LLHL | LLHL | LLHL | LLHL |
| 104-111 | LLHL | LLHL | LLHL | LLHL | LLHL | LLHL | LLHL | LLHL |
| 112-119 | LLHH | LLHH | LLHH | LLHH | LLHH | LLHH | LLHH | LLHH |
| 120-127 | LLHH | LLHH | LLHH | LHHH | LLLL | LHHH | LHHH | LLLL |
| 128-135 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 136-143 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 144-151 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 152-159 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 160-167 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 168-175 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 176-183 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 184-191 | LLLL | LLLL | LLLL | LLLL | LHHL | LLLL | LLLL | LLLL |
| 192-199 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 200-207 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 208-215 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 216-223 | LLLL | LLLL | LLLL | LHHH | LHHH | LLLL | LLHH | LHHH |
| 224-231 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 232-239 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 240-247 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 248-255 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |

WHAT WE CLAIM IS:—

1. An electronic calculator comprising:
 - keyboard input means including a plurality of alphameric keys for entering one or more BASIC language statements into the calculator, an execute control key for initiating execution of a BASIC language statement entered into the calculator, and a store control key for initiating storage of a BASIC language statement entered into the calculator;
 - memory means, coupled to said keyboard input means, for storing a program of BASIC language statements entered into the calculator;
 - processing means coupled to said keyboard input means and said memory means, said processing means being responsive to actuation of said execute control key, following entry of a BASIC language statement into the calculator, for executing that BASIC language statement, said processing means being further responsive to actuation of said store control key, following entry of a BASIC language statement into the calculator, for storing that BASIC language statement as part of a program in said memory means; and
 - output means, coupled to said memory means, for providing an output indication of the result of execution of a BASIC language statement.
2. A calculator according to claim 1 wherein:
 - said processing means includes a read-only memory storing a plurality of routines and subroutines to be selectively performed by the calculator in executing and storing one or more BASIC language statements;
 - said processing means is responsive to actuation of said execute control key, following entry of a BASIC language statement into the calculator, for selectively performing one or more of the routines and subroutines stored in said read-only memory to execute that BASIC language statement; and
 - said processing means is responsive to actuation of said store control key, following entry of a BASIC language statement into the calculator, for selectively



637

637

1,444,141

637

BIPOLAR ROM 11 OF FIGURE 81

| | | | | | | | | |
|---------|------|------|------|------|------|------|------|------|
| 000-007 | LLLL | HLLL | LLLL | LLLL | LLLL | LLLL | HLLL | HLLL |
| 008-015 | LLLL | LHLL | LLLL | HLLL | HHLL | LLLL | LLLL | LLLL |
| 016-023 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 024-031 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 032-039 | LLLL | LHLH | LHLH | LHLH | LHLH | LHLH | LHLH | LHLH |
| 040-047 | LHLH | LHLH | LHLH | LHLH | LLH | HLLH | LLH | LLH |
| 048-055 | HLLH | HLLH | HLLH | HLLH | HLLH | HLLH | HLLH | HLLH |
| 056-063 | HLLH | HLLH | HLLH | LLH | LHLH | LHLH | LHLH | LHLH |
| 064-071 | LLHL | LHHL | LHHL | LHHL | LHHL | LHHL | LHHL | LHHL |
| 072-079 | LHHL | LHHL | LHHL | LHHL | LHHL | LHHL | LHHL | LHHL |
| 080-087 | LHHH | LHHH | LHHH | LHHH | LHHH | LHHH | LHHH | LHHH |
| 088-095 | LHHH | LHHH | LHHH | LLHH | LLH | LLHH | LEHH | LHLH |
| 096-103 | LHHL | LEHL | LEHL | LEHL | LEHL | LEHL | LEHL | LEHL |
| 104-111 | LEHL | LEHL | LEHL | LEHL | LEHL | LEHL | LEHL | LEHL |
| 112-119 | LLHH | LLHH | LLHH | LLHH | LLHH | LLHH | LLHH | LLHH |
| 120-127 | LLHH | LLHH | LLHH | LHHH | LLH | LHHH | LHHH | LLH |
| 128-135 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 136-143 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 144-151 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 152-159 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 160-167 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 168-175 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 176-183 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 184-191 | LLLL | LLLL | LLLL | LLLL | LHHL | LLLL | LLLL | LLLL |
| 192-199 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 200-207 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 208-215 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 216-223 | LLLL | LLLL | LLLL | LHHH | LHHH | LLLL | LLHH | LHHH |
| 224-231 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 232-239 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 240-247 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |
| 248-255 | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL | LLLL |

WHAT WE CLAIM IS:—

1. An electronic calculator comprising:
 - 5 keyboard input means including a plurality of alphameric keys for entering one or more BASIC language statements into the calculator, an execute control key for initiating execution of a BASIC language statement entered into the calculator, and a store control key for initiating storage of a BASIC language statement entered into the calculator;
 - 10 memory means, coupled to said keyboard input means, for storing a program of BASIC language statements entered into the calculator;
 - 10 processing means coupled to said keyboard input means and said memory means, said processing means being responsive to actuation of said execute control key, following entry of a BASIC language statement into the calculator, for executing that BASIC language statement, said processing means being further responsive to actuation of said store control key, following entry of a BASIC language statement into the calculator, for storing that BASIC language statement as part of a program in said memory means; and
 - 15 output means, coupled to said memory means, for providing an output indication of the result of execution of a BASIC language statement.
- 20 2. A calculator according to claim 1 wherein:
 - 20 said processing means includes a read-only memory storing a plurality of routines and subroutines to be selectively performed by the calculator in executing and storing one or more BASIC language statements;
 - 25 said processing means is responsive to actuation of said execute control key, following entry of a BASIC language statement into the calculator, for selectively performing one or more of the routines and subroutines stored in said read-only memory to execute that BASIC language statement; and
 - 25 said processing means is responsive to actuation of said store control key, following entry of a BASIC language statement into the calculator, for selectively



performing one or more of the routines and subroutines stored in said read-only memory to store that BASIC language statement in said memory means.

3. A calculator according to either one of claims 1 and 2 wherein:

5

said keyboard input means includes a space key; and
said processing means is responsive to actuation of said space key during entry of a BASIC language statement into the calculator for introducing a blank character at selected locations within that BASIC language statement.

5

10

4. A calculator according to any one of the preceding claims wherein said output means comprises

10

output printing means, coupled to said keyboard input means and memory means, for printing lines of one or more alphameric characters each;

said keyboard input means including memory listing means for initiating the listing of any designated number of lines of one or more alphameric characters each then stored in said memory means; and

15

said processing means also being coupled to said output printing means for causing said output printing means to selectively print out the designated lines of one or more alphameric characters each in response to actuation of said memory listing means.

15

20

5. A calculator according to claim 1 wherein said keyboard input means also comprises a plurality of keys that may be associated with user-definable functions;

20

said memory means can store a program of one or more lines of one or more alphameric characters each as aforesaid and user-definable functions associated with said plurality of keys;

25

said keyboard input means including memory erase means for selectively initiating the erasure of said program and/or said user-definable functions stored in said memory means; and

25

30

said processing means is capable of selectively erasing a portion or all of said program, any one or all of said user-definable functions, or all of said program and all of said user definable functions as specified by actuation of said memory erase means.

30

35

6. A calculator according to any one of the preceding claims wherein every stored line is associated with a separate line number; said keyboard input means includes a halt execution key for designating the line number associated with a line at which execution is to be halted;

35

and said calculator further comprises a program counter, coupled to said processing means, for storing the line number associated with the line currently being executed;

40

temporary storage means, coupled to said keyboard input means and memory means, for storing the line number associated with the line at which execution is to be halted; and said processing means includes

40

45

logic means, coupled to said keyboard input means, program counter, and temporary storage means, and capable of storing the line number associated with the line at which execution is to be halted in said temporary storage means in response to actuation of said halt execution key, followed by actuation of one or more alphameric keys designating that line number, prior to execution of the lines stored in said memory means and also capable of subsequently halting the execution of those lines in response to the occurrence of a condition of equality between the contents of said program counter and said temporary storage means.

45

50

7. A calculator according to claim 6 wherein

50

said keyboard input means includes a first program control key; and said logic means is responsive to actuation of said first program control key, followed by actuation of one or more numeric keys designating the line number of a starting line of the program stored in said memory means, for initiating execution of the program stored in said memory means at that starting line.

55

8. A calculator according to claim 7 wherein:

55

said keyboard input means includes a second program control key; said memory means can store variables employed in the program stored in said memory means;

60

said logic means is responsive to actuation of said first program control key for setting all of the variables of the program stored in said memory means to an initial state prior to initiating execution of the program; and

60

65

said logic means is further responsive to actuation of said second program control key, followed by actuation of one or more numeric keys designating the line number of a line in the program stored in said memory means,

65



* C L M S *

639

1,444,141.

639

for initiating execution of the program at that line number without altering the variables stored in said memory means.

9. A calculator according to claim 1 wherein:

said keyboard input means includes an initialize key; said memory means, coupled to said keyboard input means, is capable of storing a program of one or more lines of one or more alphameric characters each entered into said calculator and of storing variables associated with that program;

and said processing means includes logic means, coupled to said keyboard input means and memory means, for setting all of the variables of the program stored in said memory means to an initial state in response to actuation of said initialize key.

10. A calculator according to claim 9 wherein:

said logic means is responsive to entry of an array definition command by actuation of said keyboard input means to store said array definition command in said memory means; and

said logic means is responsive to said array definition command, when encountered during a program execution mode of said calculator, for dedicating a portion of said memory means as storage for the previously defined array.

11. A calculator according to claim 10 wherein said logic means is responsive to actuation of said initialize key for executing an array definition command previously stored in said memory means.

12. A calculator according to claim 1 wherein every stored statement is associated with a different number;

said calculator further comprises magnetic reading and recording means, coupled to said memory means, for transferring lines of one or more alphameric characters between said memory means and a magnetic record member, said keyboard input means being operable for entering control commands into the calculator to control the operation of said magnetic reading and recording means; and said processing means includes

logic means, coupled to said keyboard input means, memory means, and magnetic reading and recording means, said logic means being responsive to one of said control commands for securing any portion of said program designated by a beginning line number and an ending line number, said logic means thereafter being responsive to another of said control commands for inhibiting transfer of the secured portion of said program from said memory means to a magnetic record member.

13. A calculator according to claim 1 wherein:

said keyboard input means includes a second plurality of keys each of which may be associated with a function of one argument,

and said processing means includes logic means, coupled to said keyboard input means and memory means, for associating a selected function with a selected one of said second plurality of keys in response to sequential actuation of selected ones of the first and said second pluralities of keys designating the selected function and the selected one of said second plurality of keys and for storing the selected function as part of the program stored in said memory means in response to actuation of said store control key.

14. A calculator according to claim 13 wherein:

said logic means is responsive to actuation of said execute control key, following actuation of a selected one of said second plurality of keys with which a selected function has previously been associated, for initiating execution of that selected function by said processing means.

15. A calculator according to claim 1 wherein:

said keyboard input means comprises a plurality of user-definable keys each of which may be associated with one or more lines of BASIC language program statements;

said memory means is capable of storing lines of BASIC language program statements associated with selected ones of said plurality of definable keys;

and said processing means includes logic means, coupled to said keyboard input means and memory means, for initiating execution of the BASIC language program statements associated with a selected one of said definable keys in response to actuation thereof.

16. A calculator according to claim 1 wherein:

said processing means is capable of defining a square matrix in response to a matrix command entered into the calculator from said keyboard input means,

654

for thereafter storing the matrix elements in said memory means as designated from said keyboard input means, and for thereafter evaluating the determinant of the previously defined square matrix in response to a determinant command entered into the calculator from said keyboard input means.

17. A calculator according to claim 16 wherein said processing means includes a read-only memory in which routines and subroutines for defining the square matrix, storing the matrix elements, and evaluating the determinant of the defined square matrix are stored.

18. A calculator according to claim 1 and further comprising a print control key for designating a print-all mode of calculator operation; and wherein said processing means includes

logic means, coupled to said keyboard input means, memory means, and display means, for initiating the display of lines of one or more alphameric characters entered into the calculator, alphameric error messages generated during execution of lines of one or more alphameric characters by said processing means, and executed display commands; and

printing means for printing lines of one or more alphameric characters, said logic means being responsive to actuation of said print control key for initiating the printing of lines of one or more alphameric characters entered into the calculator, error messages generated during execution of lines of one or more alphameric characters by said processing means, and executed display commands.

19. A calculator according to claim 1 wherein each of said alphameric keys is also representative, either alone or in combination with others of said alphameric keys, of an ASCII character;

and said calculator further comprises

buffer storage means, coupled to said keyboard input means and to said memory means, for storing a line of alphameric information entered into the calculator;

said memory means being also coupled to said buffer storage means for storing a program of one or more lines of alphameric information entered into the calculator; and said processing means being also coupled to said buffer storage means for executing the line of alphameric information then stored in said buffer storage means or a program of one or more lines of alphameric information then stored in said memory means; and said processing means includes

logic means coupled to said keyboard input means and responsive to actuation of selected ones of said plurality of alphameric keys, singly or in combination, for generating binary codes representative of every ASCII character and for storing those binary codes in said buffer storage means.

20. A calculator according to claim 19 wherein:

said keyboard input means includes a terminal transit key;

said calculator includes a modem for transmitting and receiving alphameric information over telephone lines; and

said logic means is responsive to actuation of said terminal transmit key for transmitting through said modem the ASCII codes represented by the binary codes then stored in said buffer storage means.

21. A calculator according to claim 1 and further comprising:

input/output interface means, including a plurality of input/output channels, for coupling selected peripheral input/output units to the calculator; and said processing means includes

logic means coupled to said keyboard input means, memory means, and input/output interface means, said logic means being responsive to execution of a binary output statement, including a base ten number and entered from said keyboard input means or stored in said memory means, for converting said base ten number to a corresponding twelve-bit binary code for transmitting said twelve bit binary code to a selected one of said input/output channels.

22. A calculator according to claim 1 and further comprising:

input/output interface means, including a plurality of input/output channels, for coupling selected peripheral input/output units to the calculator; and said processing means includes

logic means coupled to said keyboard input means, memory means, and input/output interface means, said logic means being responsive to execution of a binary read statement, entered from said keyboard input means or stored in said memory means, for reading an eight-bit word on a selected one of said input/output channels, for converting said eight-bit word to a base ten number, and for associating said base ten number with a designated variable.



655

23. A calculator according to claim 1 and further comprising:
input/output interface means, including a plurality of input/output
channels, for coupling selected peripheral input/output units to the calculator; and
said processing means includes

5 logic means coupled to said keyboard input means, memory means, and
input/output interface means, said logic means being responsive to execution of a
status statement, entered from said keyboard input means or stored in said
memory means, for interrogating a selected input/output channel to obtain an
10 indication of the operating state of a selected peripheral input/output unit
associated therewith.

24. A calculator according to claim 23 wherein said logic means is operable
for generating a number indicative of the operating state of the interrogated
peripheral input/output unit.

15 25. A calculator according to claim 1 wherein said processing means includes
logic means coupled to said keyboard input means and memory means,
said logic means being responsive to execution of a rotate statement, entered from
said keyboard input means or stored in said memory means, for rotating a
specified number of the individual bits of a designated sixteen-bit integer word
20 entered into the calculator or stored in said memory means.

26. A calculator according to claim 25 wherein said logic means is responsive
to execution of an AND statement for performing, according to the rules of
Boolean algebra, the logical AND operation on the corresponding bits of two
designated sixteen-bit integer words entered into the calculator or stored in said
memory means and for associating the resulting sixteen-bit word with a specified
25 variable.

27. A calculator according to claim 25 wherein said logic means is responsive
to execution of an OR statement for performing, according to the rules of Boolean
algebra, the logical OR operation on the corresponding bits of two designated
sixteen-bit integer words entered into the calculator or stored in said memory
means and for associating the resulting sixteen-bit word with a specified variable.
30

28. A calculator according to claim 1 wherein said keyboard input means
employs ASCII coding;

and said calculator further comprises
input/output interface means, including a plurality of input/output
channels, for coupling selected peripheral input/output units to the calculator; and
said processing means includes

35 logic means coupled to said keyboard input means, memory means, and
input/output interface means, said logic means being responsive to execution of an
enter statement, entered from said keyboard input means or stored in said memory
means, for receiving information from a selected peripheral input/output unit and
for converting the received information to ASCII code when it is received from a
peripheral input/output unit employing a non-ASCII code, said logic means being
40 further responsive to execution of an output statement, entered from said
keyboard input means or stored in said memory means, for transmitting
information from the calculator to a selected peripheral input/output unit and for
converting such information from ASCII code to any non-ASCII code employed
45 by the selected peripheral input/output unit.

29. A calculator according to claim 1 substantially as hereinbefore described
with reference to the accompanying drawings.

ERIC POTTER & CLARKSON,
Chartered Patent Agents,
Kingsway House,
Kingsway,
London, WC2B 6QX.

Printed for Her Majesty's Stationery Office by the Courier Press, Leamington Spa, 1976.
Published by the Patent Office, 25 Southampton Buildings, London, WC2A 1AY, from
which copies may be obtained.

