1 433 117

PATENT SPECIFICATION

(11) 1 433 117

(21) Application No. 58867/72 (22) Filed 20 Dec. 1972
(31) Convention Application No. 212 581
(32) Filed 27 Dec. 1971 in
(33) United States of America (US)
(44) Complete Specification published 22 April 1976
(51) INT CL² G06F 15/02, 15/04//9/16
(52) Index at acceptance
G4A 10EX 11C 12C 12D 15A2 15D 16D 16G 16J 17P 1D 1F 2AY 2BY 2C 2F5 2F6 2H1 3W10W2 3W4X 5A 5B 8C 9C 9X KB

(19)

(54) PROGRAMMABLE CALCULATOR

(71) We, HEWLETT-PACKARD COMPANY, 1501 Page Mill Road, Palo Alto, California, United States of America, a corporation organized 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 can be controlled both manually from a keyboard input unit and automatically by a stored programme loaded into the calculator from the keyboard input unit or an external record member. The invention is related to the invention disclosed in each of our co-pending patent applications 32755/75, 32756/75, 32757/75 (Serial Nos. 1,433,118, 1,433,119 and 1,433,120).

In the complete specification of our co-pending application No. 32755/75 (Serial No. 1,433,118), there is disclosed an electronic programmable self-contained desk-top calculator including keyboard input means having the capability of entering information in the form of numerals, algebraic expressions and alphabetic messages into the calculator; memory means for storing information entered into the calculator; processing means for processing information entered into the calculator or stored in said memory means to perform selected functions; and output means for providing an output indication of selected functions performed by the calculator; said keyboard input means including a plurality of keys which, in use, in the operation of the calculator, enter arithmetic expressions into the calculator, an execute control key which, in use, in the operation of the calculator initiates execution of an arithmetic expression entered into the calculator, a store control key which, in use, in the operation of the calculator, initiates storage of an arithmetic expression entered into the calculator, one or more keys the function of which can be defined by a user of the calculator, and means for defining a function for each definable key and storing each such function defined in said memory means; and said processing means being responsive to actuation of said execute control key, following actuation of a definable key and one or more other keys representing an arithmetic expression for executing the defined function for that definable key as a part of that arithmetic expression, and being responsive to actuation of said store control key, following actuation of a definable key and one or more other keys representing an arithmetic expression, in said memory means.

In the complete specification of our co-pending application No. 32756/75 (Serial No. 1,433,119), there is disclosed an electronic programmable self-contained desk-top calculator including keyboard input means having the capability of entering alphameric information in the form of numerals, algebraic expressions and alphabetic messages into the calculator; memory means for storing alphameric information entered into the calculator; processing means for processing alphameric information entered into the calculator or stored in said memory means to perform selected functions, and output means for providing an output indication of selected functions performed by the calculator; said keyboard input means including a plurality of keys which, in use, in the operation of the calculator enter one or more lines of one or more alphameric statements each into the calculator, and one or more flag control keys which, in use, in the operation of

the calculator, enter one or more flag statements into the calculator, each flag statement specifying a clearing, setting or testing function involving a designated one of a plurality of flag variables stored in said memory means; and said processing means being responsive to each flag statement encountered during processing of a line of one or more alphameric statements for performing the function specified by that flag statement.

In the complete specification of our co-pending application No. 32757/75 (Serial No. 1,433,120), there is disclosed an electronic programmable self-contained desk-top calculator including keyboard input means having the capability of entering lines of alphameric information in the form of numerals, algebraic expressions and alphabetic messages into the calculator; memory means for storing lines of alphameric information entered into the calculator; processing means for processing lines of alphameric information entered into the calculator or stored in said memory means to perform selected calculations; and output means for providing an output indication of selected functions performed by the calculator; said keyboard input means including a control key which, in use, in the operation of the calculator causes the calculator to operator in a trace mode of operation; and said output means comprising an output printer which operates to print the numeric results of each calculation performed by the calculator as well as each line of alphameric information entered from said keyboard input means when the calculator is operating in said trace mode.

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.

Nonprogrammable 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 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.

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 programme and data storage memory or to perform many special keyboard functions oriented toward the environment of the user.

In some conventional programmable calculators a programme 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 programmes 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 programme secure when it is recorded onto an external magnetic record member. Any user may therefore re-record the programme or obtain an indication of the individual programme steps once the programme 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 of one or more alphameric statements or an alphabetic message such as might be used, for example, to inform the user how to run programmes with which he may be unfamiliar. Such features would be very helpful to the user both in editing programmes and in simplifying their use.

Similarly, conventional programmable calculators with self-contained output printer units typically have a very limited alpha capability of only a few selected characters confined to certain columns of the printer. They are therefore typically unable to print out a distinct mnemonic representation of every programme stored within the calculator. Furthermore, they are typically unable to print out labels for inputs to and outputs from the calculator or messages informing the user how to run programmes with which he may be unfamiliar. Such features would also be very helpful to the user both in editing programmes and in simplifying their use.

In order to efficiently employ a programmable calculator to solve many non-trivial computational problems, the user must be given operator instructions that, when followed, will provide a printed listing of computed results. Since conventional calculators typically do not include both a self-contained volatile output display unit and a self-contained output printer unit, each of which has an alpha capability, the operator instructions are presented either as numbers on the volatile output display unit or as alphameric messages scattered within the printed listing of computed results. The presentation of operator instructions as numbers by the volatile output display unit is undesirable because the user must then either memorize or look up the operator instructions corresponding to the numbers displayed. Similarly, the presentation of operator instructions scattered throughout the printed listing of computed results is also undesirable.

Conventional programmable calculators typically have little or no capability for editing keyboard entries or programmes 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 programme 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 programme of one or more lines of alphameric statements. Such features would be very helpful to the user in editing programmes.

Conventional computers typically have or may be programmed to have much more capability than conventional programmable calculators. For many computational problems the computer provided little or no economical improvement over manual methods of solution because of the difficulty of the interfacing problem between the user and the machine. This interfacing problem appears as a slow turn-around time in batch processing or in a long learning period for the user due to the general-purpose nature of the computer. A skilled programmer is typically required to utilize a computer. Due to these factors, conventional computers are best suited for handling large amounts of data or solving highly iterative or very complex computational problems.

It is an object of this invention 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.

The present invention provides an electronic programmable self-contained desk-top calculator including keyboard input means having the capability of entering alphameric information in the form of numerals, algebraic (as defined) expressions and alphabetic messages into the calculator; memory means for storing information entered into the calculator; processing means for processing information entered into the calculator or stored in said memory means to perform

selected functions; and output means for providing an output indication of selected functions performed by the calculator; said keyboard input means including a plurality of keys which, in use, in the operation of the calculator enter lines of one or more alphameric statements (as defined) each into the calculator, an execute control key which, in use, in the operation of the calculator initiates execution of a line of one or more alphameric statements by said processing means, and a store control key which, in use, in the operation of the calculator initiates storage of a line of one or more alphameric statements in said memory means; and calculator further including buffer storage means coupled to said keyboard input means for storing each line of one or more alphameric statements as it is entered into the calculator for said keyboard input means; said memory means being coupled to said buffer storage means for storing lines of one or more alphameric statements each entered into the calculator; and said processing means being coupled to said keyboard input means, buffer storage means, and memory means, being responsive to actuation of said execute control key, following entry of a line of one or more alphameric statements into said buffer storage means, for executing that line of one or more alphameric statements, and being responsive to said store control key, following entry of a line of one or more alphameric statements into said buffer storage means, for storing that line of one or more alphameric statements in said memory means. The term "algebraic" where used herein is defined as relating to the abstract investigation of properties of numbers by means of symbols (a,b,-x,y,-etc). Where used herein the term "statement" means a valid instruction, or a valid combination of instructions and values. The statement is the smallest complete syntactical unit which can completely specify an activity.

Preferably said keyboard input means includes recall means for recalling a designated line of one or more alphameric statements from said memory means to said buffer storage means; and said processing means is responsive to actuation of said execute control key, following recall of a line of one or more alphameric statements to said buffer storage means, for executing that line of one or more alphameric statements, and is responsive to actuation of said store control key, following recall of a line of one or more alphameric statements into said buffer storage means, for storing that line of one or more alphameric statements into said buffer storage means.

Preferably, said output means is coupled to said buffer storage means and processing means for visually displaying a line of one or more alphameric statements stored in said buffer storage means.

Preferably, said processing means includes means for storing routines and subroutines to be performed by the calculator in executing and storing lines of one or more alphameric statements each; and means responsive to actuation of said execute control key for selectively performing one or more of said routines and subroutines to execute the line of one or more alphameric statements then stored in said buffer storage means, and responsive to actuation of said store control key for selectively performing one or more of said routines and subroutines to store the line of one or more alphameric statements then stored in said buffer storage means in said memory means.

The present invention further provides an electronic programmable self-contained desk-top calculator including keyboard input means having the capability of entering alphameric information in the form of numerals, algebraic (as defined) expressions and alphabetic messages into the calculator; memory means for storing information entered into the calculator; processing means for processing information entered into the calculator or stored in said memory means to perform selected functions; and output means for providing an output indication of selected functions performed by the calculator; said keyboard input means including a plurality of keys which, in use, in the operation of the calculator enter lines of one or more alphameric statements each into the calculator, an execute control key which, in use, in the operation of the calculator, initiates execution of a line of one or more alphameric statements by said processing means, and a store control key which, in use, in the operation of the calculator, initiates storage of a line of one or more alphameric statements in said memory means; said calculator further including buffer storage means coupled to said keyboard input means for storing each line of one or more alphameric statements as it is entered into the calculator from said keyboard input means; said memory means being coupled to said buffer storage means for storing lines of one or more alphameric statements each entered into the calculator; said processing means

being coupled to said keyboard input means, buffer storage means, and memory means, being responsive to actuation of said execute control key, following entry of a line of one or more alphameric statements into said buffer storage means, for executing that line of one or more alphameric statements, and being responsive to actuation of said store control key, following entry of a line of one or more alphameric statements into said buffer storage means, for storing that line of one or more alphameric statements in said memory means; and said output means comprising alphameric display means coupled to said buffer storage means and processing means for visually displaying a line of one or more alphameric statements stored or being stored in said buffer storage means to enable the user to check for errors therein during and after entry of that line.

In a calculator as set forth in the last preceding paragraph preferably said keyboard input means includes a first control key operable with one or more other keys, in the operation of the calculator, to designate a particular line of one or more alphameric statements stored in said memory means, and a second control key which, in use, in the operation of the calculator nondestructively recalls a designated line of one or more alphameric statements from said memory means to said buffer storage means; and said processing means is responsive to actuation of said second control key, following actuation of said first control key and one or more other keys designating a particular line of one or more alphameric statements stored in said memory means, for nondestructively transferring that line of one or more alphameric statements from said memory means to said buffer storage means to enable the user to observe a display thereof by said alphameric display means.

In a calculator as set forth in the last preceding paragraph preferably said processing means is responsive to successive actuations of said second control key for transferring successive lines of one or more alphameric statements each from said memory means to said buffer storage means to enable the user to observe a visual display thereof by said alphameric display means.

In a calculator as set forth in either one of the last two immediately preceding paragraphs preferably a separate line number is associated with each line of one or more alphameric statements stored in said memory means; and said processing means including an interpreter routine responsive to actuation of said first control key, followed by actuation of one or more other keys representing a particular line number, for computing the current address in said memory means of the line of one or more alphameric statements associated with that line number, and an uncompiler routine responsive to actuation of said second control key for nondestructively transferring the line of one or more alphameric statements specified by that address from said memory means to said buffer storage means.

In a calculator as set forth in any one of the last four immediately preceding paragraphs preferably said keyboard input means includes a plurality of editing keys which, in use, in the operation of the calculator, edit a line of one or more alphameric statements, entered into said buffer storage means from said keyboard input means or recalled thereto from said memory means, without altering the contents of said memory means; and said processing means is responsive to actuation of said store control key, following editing of a line of one or more alphameric statements stored in said buffer storage means, for storing that edited line in said memory means.

In a calculator as set forth in the last preceding paragraph preferably said processing means is responsive to actuation of said store control key, following editing of a line of one or more alphameric statements recalled to said buffer storage means from said memory means, for storing that edited recalled line back into said memory means in place of the corresponding unedited line previously stored therein.

In a calculator as set forth in either one of the last two immediately preceding paragraphs preferably said editing keys include a forward control key and a backward control key which, in use, in the operation of the calculator, single step either forward or backward through a line of one or more alphameric statements entered into said buffer storage means from said keyboard input means or recalled from said memory means; and said alphameric display means is operable for visually displaying part of all of the entered or recalled line of one or more alphameric statements up to and including as the last displayed character the last character to which that line has been single stepped.

In a calculator as set forth in the last preceding paragraph preferably said buffer storage means includes a pointer, said pointer being responsive to

actuations of either said forward control key or said backward control key for single stepping either forward or backward, respectively, through a line of one or more alphameric statements stored in said buffer storage means and for designating the last character in that line to be displayed by said alphameric display means.

In a calculator as set forth in either one of the last two immediately preceding paragraphs preferably said processing means includes routines responsive to successive actuations of said forward and backward control keys for single stepping character-by-character through a line of one or more alphameric statements recalled to said buffer storage means from said memory means and, upon reaching one end of that recalled line, for recalling the next succeeding or preceding line of one or more alphameric statements stored in said memory means to said buffer storage means.

In a calculator as set forth in any one of the last three immediately preceding paragraphs preferably said processing means includes a replacement routine for conditioning the calculator to allow replacement of one or more characters of a line of one or more alphameric statements, entered into said buffer storage means from said keyboard input means or recalled from said memory means, by actuating keys of said keyboard input means that are representative of the desired one or more replacement characters, following actuation of at least one of said forward and backward control keys for single stepping to the desired one or more characters to be replaced.

In a calculator as set forth in an any one of the last four immediately preceding paragraphs preferably said editing keys include a delete control key which, in use, in the operation of the calculator selectively delete one or more characters of a line of one or more alphameric statements entered into said buffer storage means from said keyboard input means or recalled thereto from said memory means, following actuation of at least one of said forward and backward control keys for single stepping to the one or more characters to be deleted.

In a calculator as set forth in the last preceding paragraph preferably said processing means includes a delete routine responsive to actuation of said delete control key, following actuation of at least one of said forward and backward control keys for single stepping to a particular character of a displayed line of one or more alphameric statements, for deleting that character.

In a calculator as set forth in any one of the last three paragraphs but eight preferably said keyboard input means includes a delete control key which, in use, in the operation of the calculator, delete from said memory means a line of one or more alphameric statements recalled from said memory means to said buffer storage means.

In a calculator as set forth in any one of the last seven immediately preceding paragraphs preferably said editing keys include an insert control key which, in use, in the operation of the calculator, inserts one or more characters at selected positions within a line of one or more alphameric statements, entered into said buffer storage means from said keyboard input means or recalled thereto from said memory means, following actuation of at least one of said forward and backward control keys for single stepping to the selected position.

In a calculator as set forth in the last preceding paragraph preferably said processing means includes an insert routine responsive to actuation of said insert control key, following actuation of at least one of said forward and backward control keys for single stepping to a particular position of a displayed line of one or more alphameric statements, for inserting one or more additional characters.

In a calculator as set forth in anyone of the last three paragraphs but eleven preferably said keyboard input means includes an insert control key which, in use, in the operation of the calculator, inserts one or more lines of one or more alphameric statements each between any two designated lines of one or more alphameric statements stored in said memory means.

In a calculator as set forth in the last preceding paragraph preferably said processing means includes routines responsive to actuation of said insert control key, following recall of a line of one or more alphameric statements from said memory means to said buffer storage means, for inserting one or more lines of one or more alphameric statements each into said memory means immediately following that recalled line.

In an electronic calculator as set forth in the last preceding paragraph but twenty preferably said processing means is operable, during entry of lines of one or more alphameric statements each into the calculator, for detecting syntax errors in

the entered statements as such errors occur; and said output means is responsive to detection of a syntax error by said processing means for providing a visual indication thereof to the user.

In a calculator as set forth in the last preceding paragraph preferably said processing means is operable, upon detection of a syntax error in a line of one or more alphameric statements being entered into the calculator, for inhibiting actuation of any key of said keyboard input means which is not effective in determining, removing, or correcting that error.

In a calculator as set forth in the last two immediately preceding paragraphs preferably said processing means and output means are operable for visually displaying an alphameric message indicating to the user the occurrence of each syntax error.

In a calculator as set forth in the three immediately preceding paragraphs preferably said keyboard input means includes a control key; said output means comprises alphameric display means for visually displaying alphameric information as it is being entered into the calculator; and said processing means is operable, upon detection of a syntax error during entry of a line of one or more alphameric statements into the calculator, for replacing the alphameric information then displayed by an alphameric indication of the syntax error, and is thereafter responsive to actuation of said control key for causing said alphameric display means to again display the previously replaced alphameric information, including the erroneous keyboard entry.

Alternatively, said keyboard input means includes a control key; said output means comprises alphameric display means for visually displaying alphameric information as it is being entered into the calculator; and said processing means is operable, upon detection of a syntax error during entry of a line of one or more alphameric statements into the calculator, for replacing the alphameric information then displayed by an alphameric indication of the syntax error, and is thereafter responsive to actuation of said control key for causing said alphameric display means to again display the previously replaced alphameric information up to but not including the erroneous keyboard entry.

In an electronic calculator as set forth in the last preceding paragraph but twenty-five preferably said keyboard input means includes a programme execution control key which, in use, in the operation of the calculator, initiates execution of one or more lines of alphameric information stored as a programme in said memory means; and said processing means is responsive to an error encountered during execution of a line of alphameric information entered into the calculator from said keyboard input means or stored in said memory means for providing an indication of that error to the user.

In a calculator as set forth in the last preceding paragraph preferably a separate line number is associated with each line of alphameric information stored in said memory means; and said processing means is responsive to an error encountered during execution of a line of alphameric information stored in said memory means for providing an indication of the line number associated with that line.

In a calculator as set forth in the last two immediately preceding paragraphs preferably said output means comprises alphameric display means for visually displaying to the user an indication of each error encountered during execution of a line of alphameric information and, in the case of an error encountered during execution of a line of alphameric information stored in said memory means, an indication of the line number associated with that line.

In a calculator as set forth in the last three immediately preceding paragraphs preferably said keyboard input means includes a recall control key; and said processing means in responsive to actuation of said recall control key alone or with other keys following occurrence of an error during execution of a line of alphameric information for causing said output means to display the line, or at least that portion thereof, during execution of which the error occurred.

In an electronic calculator as set forth in the last preceding paragraph but twenty-nine preferably each of said alphameric keys is associated with a different representation of the alphameric information entered into the calculator by pressing that alphameric key; said output means comprises alphameric display means, coupled to said buffer storage means, for visually displaying the representations of the alphameric information stored in said buffer storage means; and said processing means is coupled to said alphameric display means and is responsive to actuation of each alphameric key employed in entering a line of

alphameric information representing an algebraic expression and/or an alphabetic message into the calculator for causing said alphameric display means to display the mnemonic representation associated with that alphameric key.

In a calculator as set forth in the last preceding paragraph preferably said alphameric display means is capable of simultaneously visually displaying representations of an amount of alphameric information of shorter length than said buffer storage means is capable of storing; and said processing means is automatically operable, when a line of alphameric information being stored in said buffer storage means exceeds said amount of alphameric information, for causing said alphameric display means to display the representations of said amount of alphameric information last stored in said buffer storage means.

In a calculator as set forth in the last preceding paragraph preferably said keyboard input means includes one or more display control keys which, in use, in the operation of the calculator, control said alphameric display means; and said processing means is responsive to actuation of said one or more display control keys which, in use, in the operation of the calculator, cause said alphameric display means to display the representations of any desired portion of a line of alphameric information stored in said buffer storage means.

In a calculator as set forth in the last three immediately preceding paragraphs preferably said keyboard input means includes a clear control key which, in use, in the operation of the calculator, initiates the clearing of said buffer storage means and said alphameric display means; and said processing means is responsive to actuation of said clear control key for clearing said buffer storage means and said alphameric display means without affecting the alphameric information stored in said memory means.

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

In the accompanying drawings:—

Fig. 1 is a front perspective view of an adaptable programmable calculator according to the preferred embodiment of this invention.

Figs. 2A—B are a simplified block diagram of the adaptable programmable calculator of Fig. 1.

Figs. 3A—B are a memory map of the memory unit employed in the adaptable programmable calculator of Figs. 1 and 2A—B.

Fig. 4 is a detailed memory map of the dedicated portion of the data storage section of the read-write memory employed in the memory unit of Figs. 2A—B and 3A—B.

Fig. 5 is a plan view of the keyboard input unit employed in the adaptable programmable calculator of Figs. 1 and 2A—B showing how the keyboard input unit may be redefined by an alpha plug-in read-only memory module that may also be employed in the adaptable programmable calculator.

Figs. 6, 7 and 8A—C are plan view of the keyboard overlays associated with the trigonometric, peripheral control, and user definable functions plug-in read-only memory modules, respectively, that may be employed in the adaptable programmable calculator of Figs. 1 and 2A—B.

Figs. 9A and 9B illustrate the control of the calculator hardware units by firmware routines contained in the calculator ROM.

Figs. 10A—C are a simplified flow chart of the overall control sequence employed for keycode processing in the adaptable programmable calculator.

Fig. 11 is a block diagram of the memory unit of Figs. 2A—B.

Figs. 12A—D are a detailed schematic diagram of the memory address register of Figs. 2A—B and 11.

Figs. 13A—D are a detailed schematic diagram of the control circuitry of Fig. 2A—B and 11.

Figs. 14A—D are a detailed schematic diagram of the memory access register of Figs. 2A—B and 11.

Figs. 15A—D are a detailed schematic diagram of the output display unit employed in the adaptable programmable calculator of Fig. 1.

DESCRIPTION OF THE PREFERRED EMBODIMENT

GENERAL DESCRIPTION

Referring to Figure 1, 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 card reading and recording unit 14 for recording information stored within the calculator onto one or more external magnetic cards 16 and for subsequently loading the information recorded on these and other similar magnetic cards back into the calculator. The calculator also includes a solid state output display unit 18 for displaying alphameric information stored within the calculator. It may also include an output printer unit 20 for printing out alphameric information on a strip of thermal-sensitive recording paper 22.

Referring to the simplified block diagram shown in Figures 2A—B, it may be seen that the calculator also includes an input-output control unit (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 a $\overline{S}$-bus 50, a T-bus 52, and an $\overline{R}$-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 $\overline{POP}$ when power is supplied to the calculator.

The I/O control unit 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 CPU 48 via T-bus 52 or in parallel from keyboard input unit 12, magnetic card reading and recording unit 14, and peripheral input units 28 such as the marked card reader via twelve input party lines 62. Information may also be transferred from I/O register 56 either bit-serially to CPU 48 via $\overline{S}$-bus 50 or in parallel to magnetic card 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 magnetic card control circuit 67, 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 $\overline{POP}$ 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. Magnetic card control circuit 67 enables magnetic card reading and recording unit 14 to respond to information in I/O register 56 and either read information into I/O register 56 from a magnetic card 16 or record information onto a magnetic card 16 from I/O register 56. 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 $\overline{PTR}$ and $\overline{XTR}$ thereto. In response to these I/O microinstructions from CPU 48, control signal $\overline{POP}$ 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 instruction 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. RWM 78 and ROM 80 comprise MOS-type semiconductor memories. As shown in the memory map of Figures 3A—B, they are organized into nine 1,024-word pages. The basic RWM 78 contains a dedicated system storage section of 256 sixteen-bit words extending from address 1400 to address 1777 on page 0 and a separate user program and/or data storage section of 768 sixteen-bit words extending from address 16400 to address 17777 on page 7. All addresses on the memory map are represented in octal form.

An optional 1024 sixteen-bit words of RWM may be made available to the user at address 20000 to address 21777. 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 Figure 4, the RWM dedicated system storage section includes 12 words (addresses 1414—1427) employed as "X", "Y", and "Z" four-word working registers available to the user and 12 words (addresses 1400—1413) employed as "A", "B", and "C" four-word storage registers available to the user. A twenty-fifth sixteen-bit word (address 1430) contains sixteen flags which are available to the user in writing programs for the calculator. Addresses 1430—1437 comprise a system stack and input buffer area. The high address end of this area contains keycodes as they are entered from the keyboard in building a statement to the calculator. If the keycode entered has proper syntax relative to the keycodes previously entered, that keycode is then stored in the input buffer area. The low address end of the area is known as the system stack area and is employed by CPU 48 in building Polish notated algebraic strings. Fifty-seven words (addresses 1540—1626) each having a mnemonic name, contain information which is employed by the firmware routines shown in Figures 9A—B. A similar area to that just described exists at addresses 1701—1715. Another area (addresses 1627—1647) of the system RWM contains information allowing plug-in ROM modules to interface with basic system firmware. An eight word information storage area (addresses 1650—1657) is dedicated for use by the plug-in ROM modules. The following sixteen-word area is divided into two buffer areas, a display buffer (addresses 1660—1667) and I/O buffer (addresses 1670—1677). These two areas are used to store ASCII coded information to be outputted through either the display unit 18 or the printer unit 20, of Figure 1. Information to be printed is stored in the display buffer initially and is then placed on the printer by the printer firmware routines stored in the calculator ROM. ASCII characters representing information to be displayed may reside in either the I/O buffer or the display buffer. Various routines in the calculator firmware are employed for the purpose of retrieving information into ASCII characters for storage in the display or I/O buffers. Eighteen words (addresses 1716—1737 of Figure 4) are employed for the temporary storage of information as control is passed from one routine to another. 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 "Xc" and "Yc" for implemention of the trigonometric functions in conjunction with the use of the trigonometric plug-in ROM module. The word located at address 1760 is used to store the contents of one of the working registers of CPU 48 during an interrupt cycle. A variable-length "system subroutine stack" (addresses 1761—1776) is employed for storing return addresses required by programs stored in ROM 80 and as temporary storage for housekeeping information required by CPU 48. 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.

As shown in the memory map of Figure 3A—B, user program and/or data storage section of RWM 78 contains 692 words (173 registers) available to the user (as user addresses 16510—17777) for storing programs and/or data and 72 words dedicated for use by CPU 48. An additional 1024 program-step and/or data words (256 registers) may be made available to the user (as user addresses 20000—21777).

Also, as shown in the memory map of Figures 3A—B, the basic ROM 80 contains 7168 sixteen-bit words extending from address 0000 to 1377 on page 0, from address 2001 to address 7777 on page 3, and from address 16000 to address 16377 on page 7. 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 3,072 sixteen-bit words of ROM may also be added on pages 4, 5, and 6 in steps of 512 and 1,024 words. This is accomplished by simply inserting plug-in ROM modules 92 into receptacles 94 provided therefor in top panel 90 of the calculator housing as illustrated in Figure 1 by the partially-inserted plug-in ROM module on the left. As each plug-in ROM module 92 is inserted into one of these receptacles a spring-loaded door 95 at the entrance of the receptacle swings down allowing passage of the plug-in ROM module. Once the plug-in ROM module is fully inserted as illustrated by the plug-in ROM module on the right, a printed circuit terminal board 96 contained within the plug-in ROM module plugs into a mating edge connector mounted inside the calculator.

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 by momentarily interrupting power or by depressing an ERASE MEMORY key, and they are associated with definable section 91 of keyboard input unit 12 or employed to expand the functions performed by this and other sections of the keyboard input unit.

Referring again to Figures 2A—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 $\overline{S}$-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 work 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 $\overline{S}$-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 $\overline{S}$-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 $\overline{R}$-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 $\overline{R}$-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 $\overline{R}$-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 $\overline{R}$-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 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 $\overline{R}$-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 $\overline{TTX}$ and $\overline{XTR}$ 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 $\overline{S}$-bus 50 and/or from any register of register unit 114 via $\overline{R}$-bus 54 or four-bit parallel binary-coded-decimal arithmetic on data received from T-register 84 via output lines 112 taken with $\overline{S}$-bus 50 and/or from A-register 122 via output lines 134 taken with $\overline{R}$-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 and 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 82, 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 $\overline{CCO}$ 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 micro-instructions 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 $\overline{POP}$ 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.

The calculator firmware operational diagram of Figures 9A—B illustrates the basic components of the calculator firmware. These components comprises 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, while the broken lines represent information transfer between the system and user portions of RWM 78 and the firmware routines.

Referring to Figures 9A—B, 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, a keyboard input unit 12, a display unit 18, a printer unit 20, and a magnetic card reading and recording unit 14.

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 depicted on the memory map of Figures 3A—B (addresses 2127—2226), and described in detail on pages 18 and 19 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 notation of numeric output to float 9, initialize certain variables in the system RWM area for later use by other firmware routines, set up a configuration code describing which option blocks are in use, and set up a mnemonic link, a syntax link, and an execution link to the option blocks. These links are placed in RWM 78 at addresses 1627—1647 as shown in the detailed memory map of Figure 4.

After completion of the start-up routine, control is passed to the system monitor routine 202 (addresses 6221—6235), which directs all keycoards to their proper handling routines. A detailed description of this routine is shown at page 91 of the basic system firmware listing.

Figure 10A shows the sequences discussed above from "power on" through start-up to system monitor routine 202. Also shown is a more detailed drawing of the system monitor. Upon entry of the system monitor from start-up the display routine 204 (Figures 9A—B) is called. Control resides therein until a new key is depressed on keyboard 12. During the start-up initialization described above the input buffer was also given the first line of the user area of RWM 78. Thus, when the display routine is initially called the following display will appear: 0:END. This display is used to indicate the occurence of reset in machine operation.

Referring again to Figures 9A—B, control now resides in display routine 7204, after having received control from system monitor routine 202 with the command to display the input buffer. The display routine will retain control until a new key is depressed and passes control to display driver routine 206 whenever necessary to make a display. The display driver routine accepts ASCII coded

information from either the display buffer or the I/O buffer as previously described and converts such information to a 5×7 bit matrix as required by the light-emitting-diode (LED) display 18. After each complete pass through the display buffer, control is returned to display routine 204, which in turn checks a mnemonic word labeled .WMOD to see if a new key has been depressed. Upon entry a new key, control is passed to the basic interrupt routine 208, which stores the keycode information in the system RWM 78 and at the same time sets a flag indicating that a new key has been depressed. When display routine 204 discovers that a new key has been depressed, control is returned to system monitor routine 202.

Referring again to Figure 10A, it is seen that if a new key is entered the answer to the new-key question is yes, and key processing begins.

As shown in Figures 10A—B, the coding below the point labeled SMON has been implemented as a subroutine which is called by the system in processing certain keys. The processing routine for each key is determined by examining the keycode class, a biased control number CN, and a table state word TSW. TSW takes on values from one through six and uses its sign bit as a flag. For TSW equal to one, a line is being built by the keyboard. TSW equal to two indicated a current line recalled from the user RWM 78. TSW equal to three means insert a keycode rather than replace a keycode. For TSW equal to four the interpreter has control of the system. TSW equal to five indicates the intermode state. If TSW is equal to six the current line has been executed. Except for TSW equal to three the sign bit set means that the end of line character (|—) is present in the current line. For TSW equal to three the sign bit indicates overflow in the user program area. The value of TSW indicates a particular state of system monitor routine 202 shown in Figures 9A—B.

The class and control number are determined by data in MTABL, which occupies addresses 0005—0204 of ROM 80 as shown on the detailed memory map of Figures 3A—B. A detailed compilation of the information stored in MTABL relative to each key appears at pages 5—7 of the basic system firmware listing. The format for each of the sixteen-bit words in MTABL is as follows: the most significant six bits represent the control number associated with the particular keycode; the following two bits are referred to as the class; the next four bits indicate the stack priority or the option block source for the keycode; the least significant four bits show the compare priority or relative value of the keycode if it is an option block key associated with a plug-in ROM module. When a keycode is processed, the corresponding sixteen-bit word from MTABL is loaded into CODE. The keycode value is stored in SKEY; the control number is identified and placed in mnemonic location CN; and the class is extracted from the word when needed. The classes of keys are as follows: class equal to zero are the programmable keys; class equal to one are the option block keys; class equal to two are the user program interruptable keys; class equal to three are the control keys. The class two keys are the STOP, FIXED N, FLOAT N, NORMAL, TRACE, and SET-FLAG keys. The class three keys are the RUN, EXECUTE, LIST, STORE, INSERT, FORWARD, RECALL, CLEAR, DELETE, and BACK keys. The addresses of the processing routines referred to above and as shown in the flow chart of Figures 10A—C are arranged as a two dimensional array with locations determined by TSW and the biased control number CN. These addresses are stored by row. The processing routines are labeled to indicate the values of TSW and CN; that is, A.12 located on the path of the store key indicates TSW equal to one and CN equal to two. The processing routines shown in the lower portion of the flow chart indicate general considerations in processing particular keycodes, and not all routines are illustrated. The basic system firmware listing may be consulted for a more detailed explanation of the processing sequence.

Referring again to Figures 9A—B, control has resided in the system monitor routine 202 as passed to it by display routine 204 upon entry of a new key from keyboard 12. In the case of a programmable key, control is passed to the compiler routine 210. As shown in the detailed memory map of Figures 3A—B, the compiler routine resides in ROM addresses 4055—4546 and can be examined in detail at pages 50—64 of the basic system firmware listing. The compiler routine is employed to convert algebraic expressions from infix notation to Polish notation to speed program execution and also serves to check the propriety of statements entered into the calculator from the keyboard. As shown in Figures 9A—B, once processing has been completed by compiler routine 210, control is returned again to system monitor routine 202 along with syntax error information. If the current keycode is correct within the framework of previously entered keys, no syntax

error exists, and the system monitor proceeds to build ASCII information from the keycodes in the input buffer and place it into the display buffer to be displayed by display routine 204. Control will then reside in the display routine until a new key is entered.

The above described procedure continues until the end-of-line (|—) is reached, at which time the user may exercise one of several options. He may decide to store the current line as a program line within a program, in which case he depresses the STORE key, giving control to the system monitor routine 202. In response to the STORE key, information in the compiler area is brought from the system RWM area to the user RWM area and placed into the program residing therein, after which control is returned to system monitor routine 202 for building the correct display in the display buffer for use by the display routine.

If the RUN PROGRAM key is now depressed, control is passed from display routine 204 to the system monitor routine 202 and then to the interpreter routine 214, which extracts program information in Polish notation from the user RWM. Had the EXECUTE key rather than the RUN PROGRAM key been depressed, the interpreter routine would have taken the information from the compiler output area. Interpreter routine 214 occupies ROM addresses 4547—5700 as shown on the memory map of Figures 3A—B and is described in detail at pages 65—84 of the basic system firmware listing. As the interpreter routine extracts keycodes from memory their execution routines are called. The system continues in this state until a program error is encountered or a stop command is given. Keys which can effect interruption of the system during the running of a program are the so called program-interruptable keys referred to above as the class two keys. These are the STOP, FIX N, FLOAT N, NORMAL, TRACE, and SET FLAG keys.

As described above, the mnemonic word TSW is set equal to four when control resides in program interrupt routine 216. When keys are entered under this condition only those are processed which are valid at that time. If the key entered is a class two interruptable key, control is passed from the interpreter routine to the system monitor routine and then to the program interrupt routine. Upon completion of processing thereby, control returns to system monitor routine 202 and then to interpreter routine 214, provided the interruptable key depressed was not the STOP key. Upon completion of processing by the interpreter routine, control is again passed to the system monitor routine. In response to entry of the RUN PROGRAM, EXECUTE key the system monitor routine calls format routine 218 which resides in system ROM addresses 2227—2535 and is described in detail at pages 20—27 of the basic system firmware listing. The format routine converts numeric information from internal format to ASCII characters and places them in the display buffers. When such processing is complete, control is passed to the system monitor routine 202 and then to display routine 204 for display of the results.

In response to the LIST key, the system monitor routine passes control to the program list routine 220, which calls the uncompile routine 222 for the purpose of converting algebraic expressions from Polish to infix notation and placing the result in the input buffer, after which control is returned to the program list routine. Program list routine 220 then converts the information contained in the input buffer to ASCII coded information and places it in the display buffer, after which control is passed to printer driver routine 224 for the purpose of outputting sixteen characters on printer 20. Control then returns to program list routine 220 which checks whether or not more characters are required to complete outputting the line. If more characters are required, the print driver routine 224 is recalled, and program listing continues until and end-of-program mark (—1) is encountered or the STOP key is depressed on keyboard 12. When listing is complete the program list routine computes the number of data registers available to the program, adjusts the printer paper so that the listing contained thereon may be torn from printer 20, and resets the program pointers to the beginning of the program. Control is then passed to the system monitor routine 202 and the display routine 204.

Magnetic card driver 226 is called by a record and mainline routine which is part of the execution routines residing in interpreter routine 214. The record and mainline routine is given control when the interpreter routine encounters a load and record command. Magnetic card driver 226 takes data register or program information from user RWM 78 and transfers sixteen-bit words onto an external magnetic card.

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 3A—B and the basic system firmware listing.

KEY 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 card 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 Figure 1, except as otherwise indicated.

TURN-ON PROCEDURE

When the OFF/ON switch located on the front of the calculator is set to the ON position, the following display appears:

Ø: END ⊢

The calculator is then ready for operation.

INITIALIZING THE CALCULATOR

The ERASE key has the same effect as switching the calculator off and then on again. It erases all stored data and programs from memory and clears the results of any previous calculation or operation.

THE FUNDAMENTAL USER OPERATION

Communication with the calculator is through the display. In general, there are two basic steps to follow when performing operations:

1. A set of directions is written into the display by actuating the appropriate keys.

2. The calculator is then instructed to follow these directions, and the result of any numerical operation is automatically displayed. When making keyboard calculations, this step consists solely of actuating the EXECUTE key.

These two basic steps form the fundamental user operation. With a few exceptions, all operations such as making calculations, loading or running programs, giving directions to the printer, etc., consist of some variation of the fundamental user operation.

DIAGNOSTIC NOTES

In addition to displaying numbers, directions, and the results of operations, the calculator also displays diagnostic notes to inform the user of operational errors or of special situations. The basic notes are numbered from 01 to 16 (higher numbered notes are associated with the various plug-in ROM's). The note number indicates the type of error or situation. For example, NOTE 01 indicates that the calculator was given a direction which it could not understand; NOTE 16 indicates that the printer paper supply has been exhausted. A list of the basic notes and a brief description of their meanings is given in the appendix at the end of Key Operations.

When a note condition occurs in a program execution is halted. The display then indicates the note as well as the number of the program line in which the note condition occurred; e.g.,

NOTE Ø2 IN 4 -

indicates that a note 02 condition occurred during line 4.

KEYING DIRECTIONS AND NUMBERS

Directions are written into the display by actuating the appropriate keys. Suppose, for example, that the user desires to add 2 to 4 and print out the result. The keys PRINT 2 + 4 are actuated. The calculator does not, however, follow these directions until it is instructed to do so by actuating EXECUTE. It then prints (and displays) the result, 6. Numbers are keyed into the display, as on any standard office-machine, by actuating the number keys (0 through 9) and the decimal point key in the required order. If a number is negative the minus sign should be keyed first before the number is keyed. Use of commas (such as in

32,341.6) is not allowed. As is the case with a direction, even though the keyed number is displayed, it will not be executed by the calculator until the EXECUTE key is actuated. It is not normally desirable to execute just a single number. The number would usually be included within some set of directions, and then the directions would be executed.

USE OF CLEAR

The CLEAR key clears the display, but leaves the memory unaltered. It operates immediately and does not have to be followed by EXECUTE. An end-of-line symbol (|—) appears in the display when CLEAR is actuated, which indicated that the calculator is idle. It is not necessary to clear the display before keying the next direction as long as the previous direction has been executed. In this case use of CLEAR is optional. If no subsequent execution has taken place since the last direction was keyed, then CLEAR must be used.
These keys will be printed, and subsequent tracing will cease.

MAKING ARITHMETIC CALCULATIONS

For arithmetic, the fundamental user operation consists of writing an arithmetic expression into the display and then actuating the EXECUTE key, to instruct the calculator to evaluate that expression. An arithmetic expression is written into the display by pressing keys in the same order as they would be written on paper, one key per character or symbol. The arithmetic expression may then be executed by simply pressing the EXECUTE key. This is illustrated by the keying sequences and displayed answers given below.

Keying Sequence	*Displayed Answers*
3 + 6 EXECUTE	9.00
9 . 3 – 6 EXECUTE	3.30
– 7 EXECUTE	–7.00
6 * (– 7) EXECUTE	–42.00
8 . 2 5 * 4 EXECUTE	33.00
6 * 3 / (1 1 – 2) EXECUTE	2.00
$\sqrt{\ 3}$ EXECUTE	1.73
$\sqrt{\ 4 + 5}$ EXECUTE	7.00
$\sqrt{\ (4 + 5)}$ EXECUTE	3.00

As in the above examples, quantities in parentheses are treated as one quantity. Thus $\sqrt{(4+5)}$ is equivalent to $\sqrt{9}$, whereas, $\sqrt{4+5}$ adds 5 to the square root of 4. The expression 4(3+2) is the equivalent of the expression 4*(3+2). Use of the multiplication operator is implied and is therefore optional in such cases. Parentheses can be nested (i.e., parentheses inside parentheses, etc.) but they must always be balanced, that is, there must be the same number of left-handed parentheses as there are right-handed.

THE ARITHMETIC HIERARCHY.

When an arithmetic expression contains more than one operator, as do several of the preceding examples, there is a prescribed order of execution. An expression must be properly written or the answer will be wrong. The order of execution, known as the hierarchy is shown below:

1. Mathematical functions such as square root;
2. Implied multiplication;
3. Multiplication and division; and
4. Addition and subtraction.

Where an expression contains two or more operators at the same level in the

hierarchy, they will be executed in order from left to right. The use of parentheses enables the order of execution to be changed. Thus, in the expression $\sqrt{(4+5)}$ the addition operator is executed before the square root operator even though the addition operator occupies a lower level in the hierarchy.

EXCEEDING THE LENGTH OF THE DISPLAY.

The length of an expression is not limited to the length of the display. As each excess symbol is keyed, the display shifts left to make room. The maximum allowable length for an expression varies between 35 and 69 keystrokes, depending upon the nature of the expression. If too many keys are pressed the display shows NOTE 09 (see the section on diagnostic notes below). Depending upon the nature of the expression the note may appear either before or after the EXECUTE key is pressed. In either case, the operator must press CLEAR and write a shorter expression.

MAKING CORRECTIONS.

The BACK and FORWARD keys enable a displayed expression to be altered or corrected without re-keying the entire sequence. If a wrong key is pressed when writing an expression, it can be corrected immediately by pressing the BACK key followed by the correct key, as illustrated below:

Keying Sequence	Display
2 + BACK * 4	2 * 4

A displayed expression can be blanked, key by key in reverse order, by pressing BACK one for each displayed key. The blanked keys can then be returned to the display one at a time by pressing FORWARD. If an expression contains a wrong key, press BACK until that key is blanked, press the correct key and then press FORWARD to return each subsequent key (or, if extra keystrokes are required, key in the remainder of the expression). For example, if the number 123456789 is keyed incorrectly into the display as 123444789, the error may be corrected as indicated by the following steps:

Keying Sequence	Display
BACK BACK BACK BACK BACK	1234
5 6 FORWARD FORWARD FORWARD	123456789

If the incorrect expression has been executed but no key has since been pressed, the expression can be returned to the display (by pressing BACK), corrected as before, and then again executed.

Any line of a stored program may be recalled into the display and then completely blanked by repeatedly actuating the BACK key. One additional actuation of the BACK key will bring the entire next preceding line of the stored program into the display. It is then possible to backstep through that line and bring its predecessor into the display, etc. Analogously, the FORWARD key may be repeatedly actuated to bring those lines succeeding the current line into the display.

To remove a portion of a line the BACK key is repeatedly actuated until the right most character, symbol or mnemonic of the portion to be deleted becomes the right most item in the display. The DELETE key is then actuated once for each character, symbol or mnemonic to be removed. Then, if the right most item of the line is not visible in the display, the FORWARD key is repeatedly actuated. The user may then continue writing the line, execute it, or store it, as appropriate. For example, assume it is desired to delete the underlined portion from the following line:

FXD 2;X→Y;PRT (A+B)/A;GTO 4

This is accomplished by repeatedly actuating the BACK key until the display appears as follows:

;X→Y;PRT (A+B)/A

Next, the DELETE key is actuated thirteen times. At first the display shifts to the right to bring the first part of the line into view, which in this case is FXD 2. However, FXD will not appear until there is room in the display for all four characters plus the space between D and 2. After this first part of the line comes into view, the line appears to shorten by losing an item from the right-hand side of the display each time the DELETE key is actuated, while the rest of the line remains stationary. After the segment has been deleted, the FORWARD key is repeatedly actuated until the end of the now modified line comes into view as follows:

FXD 2;GTO 4

The user may now continue writing this line, execute it, or store it, as he desires.

To add a segment to the interior of a line the BACK key is repeatedly actuated until the right most item visible in the display is the character, symbol or mnemonic immediately preceding the segment sought to be added. The INSERT key is then actuated and followed by the keys which describe the desired segment. The FORWARD key is next repeatedly actuated until the end of the line is in view. As the keys following INSERT but preceding FORWARD are actuated their mnemonics are inserted into the line with no loss of any other items in the line. The right-hand portion of the line is shifted to the right to make room for the additional items being inserted. This action continues until one of the keys, BACK, FORWARD, DELETE, CLEAR, EXECUTE or STORE is actuated. Generally the insertion of a portion of a line is terminated with the FORWARD key to return to the end of the line. For example, assume it is desired to insert the portion

20→B

into the line

10→A;30→C

To accomplish the insertion, the BACK key is repeatedly actuated until the semicolon becomes the right most item in the display. The INSERT key is then actuated and followed by the key sequence 20→B. Next, the FORWARD key is actuated until the entire line is visible as follows:

10→A,20→B;30→C

If an error is made by the user during the entry of a portion of a line being inserted into an existing line, the erroneous items may be removed by actuating the DELETE key. The user may then continue writing the desired line portion after actuating the INSERT key.

In addition to modifying individual lines of a program as discussed above, it is also possible to insert entire lines into or delete entire lines from, the interior of a program stored in memory. If it is desired to add a line between existing lines 4 and 5, the added line would become new line 5 while the old line 5 would become the new line 6. Similarly, if it is desired to remove line 3 from a program, the old line 4 would become the new line 3, the old line 5 would become the new line 4, etc. In both cases the number of available R registers is automatically adjusted after the change.

To insert a line into a program the program line counter is first set to the line number which will be associated with the new line. This may be accomplished, for example, by actuating the GO TO key followed by the number keys representing the line number followed by the EXECUTE key. The new line is then written into the display and followed by sequential actuation of the INSERT and STORE keys. The new line becomes stored, and all succeeding lines of the program together with their line numbers are shifted to provide room.

To delete a line from a program the program line counter is first set to the line number of the line to be deleted. Sequential actuation of the RECALL and DELETE keys will remove the line and shift all succeeding lines and their line numbers to close the gap.

THE DATA MEMORY.

The basic calculator contains 179 registers: six storage and working registers

(A, B, C, X, Y and Z) and 173 program and data storage registers (R0 through R172). An additional 256 R-registers (R173 through R428) may be added giving a total of 435 registers.

The A, B, C, X, Y and Z registers are selected by pressing the A, B, C, X, Y and Z keys, respectively, while the R registers are selected by pressing the R( ) key followed by the appropriate number keys 0 through 172 or 428. The argument of the R( ) key may be a computed quantity. For example, sequentially pressing the R( ), (, 7, 0, /, 2, and)keys denotes the R35 register. The argument of the R( ) may also be a variable. Then, if register A contains the number 15, sequentially pressing the R( ) and A keys denotes R15 register. Similarly, if the R5 register contains the number 10 and the C register contains the number 25, sequentially pressing the R( ), (, R( ), 5, +, C, and) denotes the R35 register.

The register denoted by the keying sequence R( ), R( ), R( ) . . . R( ) followed by one or more number keys is determined by the number designated by the number keys and by the numbers contained in the various registers. For example, the keying sequence R( ), R( ), 2 denotes the R8 register if R2 contains the number 8.

When the number following the R( ) key does not have a strictly integral value, the fractional part of the value is ignored. Thus, the keying sequence R( ), 3, 5, 6, ., 6 denotes the R35 register. A plus sign immediately following the R( ) key is dropped when the line containing it is stored. Thus, the keying sequence R( ), +, /, % is stored as R( ) 35. A minus sign immediately following the R( ) key is not permitted, and causes a syntax error (NOTE Ø1). If the R( ) key is followed by a quantity whose value is either negative, or greater than the number of available R registers, an error during execution results (the indication will be either NOTE Ø5 or NOTE Ø6, depending upon the exact circumstances).

Some of the plug-in read-only memory modules require part of the memory for their own use. When one of these modules is installed, it automatically takes the required registers, starting at the highest numbered register and working downwards. Those registers are then temporarily not available for program or data storage, until the module is removed.

When programs are stored they start in the highest-numbered available R-register and sequentially fill the memory downwards. Programs cannot be stored in the A, B, C, X, Y and Z registers. It is, therefore, most convenient to store data first in the A, B, C, X, Y and Z registers and then in the lower numbered R-registers. If the memory contains no program (i.e. at turn-on, or if ERASE has been pressed), then all registers (except those required by a plug-in read-only memory module will be available for data storage. If the memory does contain a program, then the higher-numbered registers will not be available for data diagnostic NOTE Ø6 will be displayed if the operator attempts to store data in a register which is not available.

The number of available R-registers can be determined at any time by pressing CLEAR LIST STOP. The printer will start to list the program (the STOP saves having to wait for the whole program to be listed). At the bottom of the list will be a number preceded by the letter R indicating the number of R-registers available. (The lowest-numbered register is R0; subtract 1 from the number printed to obtain the name of the highest-numbered register available for data storage).

STORING DATA.

One register can contain one data-number. It is not necessary to clear a register before storing a number in it because the number being stored automatically substitutes for the existing stored number. The entire memory is, however, cleared at turn-on or if ERASE is pressed. Storing data requires use of the → key. For example, pressing

1 2 . 6 → A EXECUTE

stores 12.6 in the A register. Similarly, pressing

6 → X EXECUTE

stores 6 in the X register, and pressing

1 9 → R( ) 1 2 EXECUTE

stores 19 in register R12. A stored number may be viewed by using either the DISPLAY or the PRINT keys. For example, pressing

DISPLAY A EXECUTE

displays the number currently stored in A (the number remains stored in A). Similarly, pressing

PRINT R( ) 1 2 EXECUTE

prints the contents of R12 (the number remains stored in R12).

IMPLIED Z.

In general, if a stored number is to be kept for any length of time it should not be stored into the Z register because the result of any arithmetic expression is automatically stored in Z if no other storage location is specified, thus

1 4 . 2 EXECUTE

is equivalent to

1 4 . 2 → Z EXECUTE

Both expressions result in a display 14.2 which is also stored in the Z register. Similarly,

3 * 4 + 1 6 / 3 EXECUTE

is equivalent to

3 * 4 + 1 6 / 3 → Z EXECUTE

A statement involving numerical activity usually contains an instruction, such as PRT, DSP, or →. If there is no such instruction, the form <quantity> →Z; or <mathematical expression> →Z, is usually automatically assumed when the line is executed or stored.

The automatic addition of Z onto the end of a statement is called the 'implied store in Z'.

For instance, if the operator presses A EXECUTE to view the contents of A, the line A→Z is what is actually executed. The contents of A are seen because that is the numerical quantity associated with the last assignment instruction executed in the line. Meanwhile, the contents of Z have been replaced by those of A, and are lost. The recommended procedure for viewing the contents of a register is to use the PRINT or DISPLAY statements, as they do not disturb the contents of any registers.

Because of the implied store into Z, the Z register is not recommended for storing data during calculations performed from the keyboard, except in certain situations. For instance, suppose the operator wished to add a series of numbers: n_1, n_2, n_3, . . . To do this, the register is first set to zero by executing the line 0→Z. Then, the numbers are added in the following manner:

$n_1 + Z$

$n_2 + Z$

$n_3 + Z$

Because of the implied store into Z, this is what is actually happening:

$n_1 + Z \rightarrow Z$	$n_1 + 0 \rightarrow Z$
$n_2 + Z \rightarrow Z$	$n_2 + n_1 \rightarrow Z$
$n_3 + Z \rightarrow Z$	$n_3 + (n_1 + n_2) \rightarrow Z$
.	.
.	.
.	.

REGISTER ARITHMETIC.

Arithmetic expressions may be written using register names instead of actual numbers. When the expression is executed, the values currently stored in those registers will be automatically substituted for the register names in order to evaluate the expression. For example, assume the user has made the following storage assignments:

12.6 in A

6 in X

19 in R12

With the above values stored, the keying sequence

A + R( ) 1 2 – X EXECUTE

would be equivalent to the keying sequence

1 2 . 6 + 1 9 – 6 EXECUTE

Other values stored in these registers would, of course, give a different result for the same expression.

Numbers and register-names may be mixed in an expression, as follows:

3 * 1 2 . 6 + 4 – 6 EXECUTE

FIXED- AND FLOATING-POINT NUMBERS.

Numbers can be keyed into the display and displayed in either fixed point or floating point notation. In fixed-point notation, a number appears in the display as commonly written, with the decimal point correctly located. Floating-point numbers are written with the decimal point immediately following the first digit (discounting leading zeros) and with an exponent. The exponent, which represents a positive or negative power of ten, indicates the direction, and the number of places, that the decimal point should be moved, to express the number as a fixed-point number. In the calculator the exponent may be an integer within the range –99 to +99. Examples of fixed point and floating point notation follow:

Fixed	Floating
1234.5	$= 1.2345 \times 10^{3}$
0.0012345	$= 1.2345 \times 10^{-3}$ (exponent)
1.2345	$= 1.2345 \times 10^{0}$

The FIXED N key selects fixed point display of displayed results. The letter N indicates that the key must be followed by one of the number keys (0 through 9) to select the number of digits to be displayed to the right of the decimal point.

The FLOAT IN key operates in the same way as FIXED N except that floating point display is selected, with N designating the required power of ten.

(When the calculator is turned on, FLOAT 9 is automatically assumed.). For example, the number 123.456789 in float 9 notation would be displayed as 1.234567890E0Z. The letter E in the display indicates that the next two digits constitute the exponent. If the exponent is negative a minus sign follows the E, as illustrated below.

Keying Sequence	Display
0 0 1 2 3 4 EXECUTE	1.234000000E–03

No more than ten significant digits can be displayed; therefore if a number becomes too large to be properly displayed as a fixed point number, it will be automatically displayed as a floating point number. If the number becomes too small, only zeros are displayed but the number may still be seen if floating point notation is then selected.

The ENTER EXPONENT key is used to designate the E (exponent) when numbers are being keyed in floating point form, as illustrated below:

Keying Sequence	Display
FLOAT N 4 EXECUTE	
2 . 5 6 ENTER EXP 2 EXECUTE	2.5600E 02
4 . 7 3 ENTER EXP — 2 EXECUTE	4.7300E–02

RANGE OF CALCULATION.

The range of the calculator is from $\pm 10^{-99}$ to $\pm 9.999999999 \times 10^{99}$; when this range is exceeded during a calculation diagnostic NOTE 10 is displayed. Calculations which normally result in zero, such as subtracting a number from a number equal to itself, do not exceed the range.

OPERATING THE PRINTER.

The print key is used to print both numerical values and alphameric messages (the form of a numerical printout is changed by the FIXED N and FLOAT N keys in the same way as the display is changed). This is illustrated by the following examples (in which it is assumed the FIXED N key, 2 key and EXECUTE key have previously been pressed to determine the form of the printout):

Printing Operation	Keying Sequence	Printout
Print A Number	PRINT 1 2 3 EXECUTE	123.00
Print result of a calculation		
	PRINT 6 + 8 / 2 EXECUTE	10.00
Print contents of a storage register		
	PRINT A EXECUTE	(CONTENTS OF A)

To print an alphanumeric message requires the use of the quote (") to both start and end the message (the quote symbol is not printed) as illustrated by the following example:

Keying Sequence

PRINT " M E S S A C E SPACE N O . 2 " EXECUTE

Printout

MESSAGE NO. 2

no more than sixteen characters (including spaces) can be printed on one line of a message, and each line must be enclosed in quotes. When following the same

PRINT instruction, lines must be separated by commas, as indicated below:

PRINT "--------" , "--------" EXECUTE

This prints two lines. If messages and values are to be mixed, they must be separated by a comma as illustrated by the following example in which it is assumed that the number 456 has been stored in the A register.

PRINT " A = " , A EXECUTE A=456.00

Pressing the SPACE N key followed by one or more number keys designating any one of the numbers 0 through 15 causes the printer to space vertically (the number key specified in the number of lines spaced). This is illustrated by the following example:

Keying Sequence	Printout
PRINT " D A Y S " EXECUTE	DAYS
SPACE N 2 EXECUTE	
PRINT 4 EXECUTE	4.00

When used in a message, most keys result in the character printed being the same as the character on the key. The following keys are the exceptions:

1. SPACE prints one blank character-space
2. GO TO prints @
3. R( ) prints :
4. STOP prints !
5. ENTER EXP prints ↑

The following keys either cannot be used in a message or they result in some meaningless character being printed:

1. All of the half-keys at the top of the keyboard and the four blank keys in the left-hand keyblock.

2. The EXECUTE key, RUN PROGRAM key, and STORE key.

3. The JUMP key, END key, IF key, GO TO/SUB key, FLAG N key, RETURN key, and SET/CLEAR FLAG N key.

PROGRAMS.

A program enables the calculator to automatically execute the keys necessary to solve a particular problem. First the program must be loaded into the calculator's memory to teach the calculator which key sequences are required and the order in which they are to be executed. Once loaded, the calculator can remember that program until a new one is loaded over it or until the calculator is switched off. A program need not be keyed into the calculator more than once because a loaded program can be recorded on magnetic cards. Recorded programs may then be loaded back into the calculator any time in the future. Once the program has been loaded, it is initialized, and then execution is commenced by actuating RUN PROGRAM key.

A complete program consists of lines of program information, each of which may be separately loaded into the calculator memory from the keyboard by actuating the STORE key when the line has been completed. An end-of-line symbol ⊢ is automatically displayed at the end of each line after that line has been stored. A program line counter keeps track of which line of a program is currently being executed or is about to be executed or stored next. Before storing a line into

the calculator memory, it may be edited with the aid of the BACK, FORWARD, CLEAR, DELETE and INSERT keys. After all lines of the program have been stored, individual lines may be recalled into the display for editing or other purposes. Recall is accomplished by sequentially actuating the CLEAR and GO TO keys followed by the number keys representing the line number of the line to be recalled followed, finally, by the RECALL key. When restoring the recalled line or the edited version thereof it is only necessary to actuate the STORE key.

MAGNETIC PROGRAM CARDS.

A magnetic card 16 such as that shown in Figure 1 is used to permanently or temporarily store programs or data. The card has two sides that may be used independently to store either data or programs (however, data and programs cannot be mixed on the same side of the card). Once a recording has been made on a card-side, that card-side can be protected from erasure by tearing out a corresponding protect tab on the card. The recording on a protected card side cannot be changed.

A program loaded into the memory may be recorded on a magnetic card 16 by pressing

END EXECUTE RECORD EXECUTE

to start the card-reader motor and by then inserting an unprotected card into the card reader. The program from the card may be loaded back into the memory by first sequentially pressing the ERASE key to clear the memory, by then pressing the END, EXECUTE, LOAD and EXECUTE keys, and by thereupon inserting the card into the card reader.

THE PROGRAM LINE.

Even though the lines of a program are stored in the same memory as data, the length of individual lines bears no relationship to the length of a register. The calculator simply uses however many registers are necessary to accommodate a particular line. The length of a line is determined by the programmer and depends upon the requirements of his program. However, the length is limited by machine requirements, in the same way that an individual expression is limited (see Exceeding the Length of the Display). Diagnostic NOTE 09 appears either before or after STORE is pressed, if the line is too long. When NOTE 09 appears the operator should press CLEAR and key in a completely new (shortened) line.

Line numbers are automatically assigned, by the calculator, in strict numerical sequence, beginning with line 0. The operator must know what line numbers will be assigned if there are any GO TO statements in his program. The line numbers are not strictly a part of the program because they will automatically change if the program is moved to a different location in memory. For example, suppose a program (No. 1) is a ten-line program (lines 0 through 9) and is already stored in the memory. If a second program (no. 2) is now loaded below program No. 1, then the first line of program No. 2 will be line 10, whereas, if program No. 2 had been the only program in the memory, then its first line would have been line 0. (Any GO TO statements must be corrected, by the programmer, to reflect any such line number changes).

A line can have one or more statements, separated by semi-colons. The actual number of statements on any one line is generally not significant, it being more important to have the statements in the correct order rather than on a particular line. Position of a statement does become significant where a line contains an IF statement or where a branch is to be made. In the former case, those statements which are to be conditionally executed must be on the same line as the IF statement and must come after the IF. In the latter case, a branch is always made to the beginning of a line. Therefore, the first statement to be executed after a branch must be the first statement of the line to which the branch is made. It is recommended that not too many statements be put on one line because a short line is easier to change (once stored) than a long line.

THE DATA ENTRY STATEMENT.

Program statements resulting from actuations of the ENTER key are used to halt the program during execution so that the user can key in data. The simplest statement contains only a register name, which is displayed when program execution is halted. The data keyed during the halt is stored, into the register

designated, when RUN PROGRAM is subsequently pressed. For example, ENT A; results in the keyed data being stored in register A. An enter statement may contain several register names (which must be separated by commas). The program will halt for each register in turn. For example, ENT A, R13, X; is the equivalent of the three separate statements ENT A ENT R13; ENT X;. A label (followed by a comma) may precede the register name. In this case the label will be displayed, instead of the name, when the halt occurs. For example, ENT "A=?",A; displays A=? and stores the subsequent data entry into register A.

BRANCHING.

Program lines are normally executed in numerical sequence. However, some statements cause the sequence of execution to be changed. This is known as branching (instead of the program going to the next sequential line, it branches to some other specified line and continues program execution there). There are two kinds of branching, conditional and unconditional. Unconditional branching is accomplished with the GO TO, JUMP and GO TO SUB keys while conditional branching is done with the IF key.

There are three types of unconditional branching with GO TO. The first type is an absolute GO TO. On absolute GO TO statements take the form GO TO N, where N is an integer that refers to a particular program line. The second type is a relative GO TO. The form of the relative GO TO statement is GO TO + N or GO TO – N, where N is an integer. This means to skip forward or backward N program lines. The third type is a GO TO label. This type of GO TO statement takes the form GO TO "LABEL", where LABEL is any unique alphameric group of characters and must be enclosed in quotes. The number of characters in the label is virtually unlimited, however, the calculator will only look at the last four characters in the label. When a GO TO "LABEL" statement is executed the program will branch to a program line with "LABEL" as the first statement of that line, where LABEL has the identical last four characters as the original GO TO "LABEL" statement. If two lines have the same label branch execution will always go to the first label.

In a program, a GO TO statement causes program execution to continue with the line whose number is specified. When a GO TO statement is entered from the keyboard and followed by the RUN PROGRAM key, the GO TO statement causes program execution to start at the line whose number is specified. However, when a GO TO statement is entered from the keyboard and followed by the EXECUTE key, the GO TO statement causes the calculator to go to the line specified but not to start program execution. Any subsequent activity then depends upon the next key pressed. A line number is valid only if a currently stored program has a line identified by that number, or if it is the next higher number after the number identifying the last stored line. All other numbers are non-valid and, if used in a GO TO statement, will cause diagnostic NOTE 08 to be displayed.

JUMP allows relative branching. But, unlike the GO TO, can have a numeric constant, a register or any legitimate calculator expression as a parameter. JUMP-6 on execution would go back six lines in the program. If the contents of A were 6.23 then JUMP A would jump the integer value of A lines, or in this case 6 lines in the program. If A were 6.23 and B were 2, then JMP (A + B) would be acceptable and would jump eight lines on execution.

Often it is desirable to execute the same operations at several places in a program. One could simply repeat a group of program lines as needed, but this can be time consuming and error prone. More important, unnecessary repetition of program lines wastes memory space. The calculator has the capability to store a set of program lines once, and allow a program to execute this set of lines many times. Such a group of program lines is called a subroutine.

Once a subroutine has been written and stored in memory, execution may branch to the subroutine from a program. This is known as calling a subroutine. The program which calls the subroutine is usually referred to as the mainline program or calling program. When the subroutine execution is completed a branch is made back to the calling program and mainline execution is resumed where it was interrupted by the subroutine call. The branch from the subroutine to the mainline program is called a return. Note that if a subroutine is called in line N; the return is made to line N + 1.

Branching to a subroutine is accomplished by using the GO TO SUB. GO TO SUB works almost exactly like GO TO and may branch to an absolute, relative or

"LABEL" address. The difference between GO TO and GO TO SUB is that when a GO TO SUB is used for a branch, the calculator stores the line number for the return branch address. To make the return branch RETURN is stored at the end of the subroutine. The calculator itself will provide the address for the return branch.

The IF statement allows the powerful feature of conditional branching in the calculator enabling the calculator to decide whether or not to execute the succeeding statement(s) on the same line as that IF statement. The general form of the IF statement is IF followed by a condition completing the statement. (For Example, IF A–B;). The line in which the IF statement appears may be completed with any other statements. The operation will be as follows. First the condition immediately following the IF will be evaluated to check the truth of the condition. If the condition is true, the statements following the IF statement are executed, and if the condition is false, execution immediately goes to the next line. Thus, in the example given above, A = B is first computed to determine whether the contents of the A register equal the contents of the B register. If this condition is true, the rest of the line would be executed. If it is false, the rest of the line would be ignored and execution would go immediately to the next line.

The conditions in IF statements all use one of the following keys to test the relationship of any two values, registers, arithmetic expressions, or flags:

1. > (greater than)
2. ≤ (less than or equal to)
3. = (equal to)
4. ≠ (not equal to)

If the relationship is the same as that indicated by the key used an asnwer of true (one) will be given and if not an answer of false (zero) will be given. For example, if the contents of A and B were 2 then

$$A = B \rightarrow C$$

would store 1 in C,

$$A \neq B \rightarrow C$$

would store 0 in C, and

$$A + B = A \rightarrow C$$

would store 0 in C.

Again, these can be used in any expression A + B (A = B) + AB (A ≤ B) + (A + B + C) (A > B) → C would store 2 + 2(1) + 4(1) + 6(0) which is 8 in C.

THE STOP AND END STATEMENTS.

The STOP key, used as a statement in a program or pressed while a program is running, halts program execution. STOP should be used only to abort a program (in the sense that it is no longer desired to run the program, or that it is desired to start execution again at the beginning).

The END key serves the dual purpose of halting program execution and of initializing the calculator for commencing program execution at line 0.

THE FLAGS.

The calculator makes sixteen flags available to the user as selected by the FLAG N key followed by numeric keys to designate one of the flags 0 through 15. For example, actuation of the FLAG N 4 selects flag 4. Flags are used generally as part of an IF statement to enable the user to define some special condition.

The calculator terminology used to describe flags is quite simple: If a flag is raised, it is set; a set flag is considered to have the value 1. If a flag is lowered, it is cleared; a cleared flag is considered to have the value 0.

Flags are set and cleared by means of the SET/CLEAR FLAG N key. This

key is actuated once to set a flag and twice to clear it. For example, a single actuation of the SET/CLEAR FLAG N key followed by the 1 and 2 number keys sets flag 12. Similarly, a double actuation of the SET/CLEAR FLAG N key followed by the 7 key cleared flag 7. Once set, a flag remains set until it is deliberately cleared. However, all flags are automatically cleared at turn-on, or when ERASE is pressed, or when and END statement is executed.

As long as no program is being executed, the state of any flag can be examined actuating the FLAG N key followed by number keys representing the flag in question followed by the EXECUTE key. The state (value) of the flag will then be displayed. Such a test will not change the state of any flag.

In addition to their normal use, flags 0 and 13 also have a special purpose. Flag 0 may be set from the keyboard while a program is actually running, by pressing the SET/CLEAR FLAG N key. Flag 13 is set automatically if the program halts for an ENTER statement and the RUN PROGRAM key is then actuated without any data being keyed.

LIST MODE.

The LIST key facilitates printing by means of the calculator printing unit a program listing of an internally stored program. The listing includes the line number of each line together with an alphameric mnemonic representation of the line. An indication of the number of storage registers remaining is printed at the end of the listing.

Program listing is accomplished by first setting the program line counter to the line at which listing is to commence. This may be done by actuating the GO TO key followed by the number keys representing the line number followed by the EXECUTE key. Next, the LIST key is actuated to begin the listing operation, which will terminate at the last program line stored.

TRACE MODE.

A trace mode of the calculator enables the user to obtain a printed record of its operation. The form of this printed record is a function of the type of operation in progress.

The calculator may be placed in the trace mode by actuating the TRACE key followed by the EXECUTE key or by program execution of a TRACE command. The calculator may be returned to normal mode by actuating the NORMAL key followed by the EXECUTE key or by program execution of a NORMAL command. The calculator is automatically placed in the normal mode when it is turned on.

While in the trace mode, the calculator prints a representation of each line execution from the keyboard and the results of those executed statements which produce a quantity that is considered a result. A few keys, such as CLEAR, are not printed.

The following example is illustrative of the printout obtained when the calculator is operating in the trace mode:

```
Ø→A;Ø→B|—
                    Ø.ØØ
                    Ø.ØØ
A+1→A;B+1Ø→B|—
                    1.ØØ
                   1Ø.ØØ
A+1→A;B+1Ø→B|—
                    2.ØØ
                   2Ø.ØØ
PRT "A=",A,"B=",
B|—
A=                  2.ØØ
B=                  2.ØØ
```

While running a program in the trace mode the calculator prints the line number of each line as it is executed, and below that, any quantities that were stored into registers by that line. Running a program in the trace mode may be very helpful in debugging a program by analyzing the numbers stored during the execution of the program. A program may, without alteration, be run in the trace mode simply by sequentially actuating the TRACE and EXECUTE keys before the execution of the program is begun. In addition, the calculator may be placed in the trace mode during execution of any program which does not contain a NORMAL statement by simply actuating the TRACE key. It is not necessary to halt execution of the program first.

DIAGNOSTIC NOTES APPENDIX.

The following diagnostic notes and associated explanations will be displayed when a particular program or operator error is detected by the calculator.

NOTE Ø1: In view of the preceding keys, the last key pressed does not make sense to the calculator. For example, a multiplication operator following the R( ) key. Note 01 is the most commonly seen note and generally occurs as soon as an incorrect key is pressed.

NOTE Ø2: An attempt to execute an instruction which is followed by an improper value; for example, the FIX N key followed by a number larger than 9.
Taking a square root is a special case:
a. $\sqrt{-}$ causes NOTE 01 when minus is pressed.
b. $\sqrt{(-4)}$ or $\sqrt{A}$ (where A contains a negative number when executed cause NOTE 02 to appear.

NOTE Ø3: Statement has an extra left-hand parenthesis [(] or a missing right-hand parenthesis [)].

NOTE Ø4: Statement has an extra right-hand parenthesis [)] or a missing left-hand parenthesis [(].

NOTE Ø5: a. Attempt to use a non-existent or unavailable R-register as a value in an expression. b. Attempt to designate a flag other than as an integer from 0 through 15.

NOTE Ø6: a. Attempt to store into a non-existent or unavailable R-register. b. Attempt to enter a number whose exponent has an absolute value greater than 99.

NOTE Ø7: Attempt to execute a RET not preceded by a matching GSB.

NOTE Ø8: Attempt to execute a GTO followed by an invalid line number or label. Also appled to GSB and JMP.

NOTE Ø9: a. Writing, executing or storing too long an expression or program line. b. Nesting subroutines too deeply.

NOTE 1Ø: An intermediate or final result of a calculation exceeds the range of the calculator.

NOTE 11: a. Pressing any half-key in the three left-hand keyblocks when:
1) It is not part of a quote field; e.g. PRT"..." and
2) The key is not defined by some plug-in read-only memory module.
b. Attempt to execute an ENTER statement from the keyboard instead of in a program.

NOTE 12: a. Storing a program line [or loading a program or data from a magnetic card] and exceeding the memory.
b. No GTO or GSB preceding LOD when loading a program (from a magnetic card) under the control of the existing program.

NOTE 13: Attempt to record on a protected magnetic card.

NOTE 14: An additional card side is required when recording on, or loading from, a magnetic card. Press EXECUTE and insert the next card-side.

NOTE 15: Appearing after a program has been loaded from a magnetic card, indicates that the calculator does not have the same ROM's installed (in the same slots) as it did when the card (or cards) was recorded. This will not affect the running of the program as long as the particular ROM's required for that program are installed in the same slots (press CLEAR and run the program in the normal way).
Recordings made when no ROM's are installed do not result in NOTE 15 when they are loaded into calculators which do have ROM's installed.

NOTE 16: Attempt to use the printer when paper supply has been exhausted. To

continue using the calculator without printer paper: If the PRINT instruction came from the keyboard, press CLEAR; if from the program press STOP RUN-PROGRAM.

SYMBOLS AND MNEMONICS.

The table below shows the symbols and mnemonics for the keys of the calculator as they are used both inside and outside of a quote field (some keys have two different symbolics or mnemonics under these two different conditions).

(42)

SYMBOLS AND MNEMONICS
FOR THE
KEYS OF THE CALCULATOR

KEY	IN QUOTE FIELD? NO	YES	KEY	IN QUOTE FIELD? NO	YES
1	1	1	R( )	R	≡
2	2	2	=	=	=
3	3	3	≠	≠	≠
4	4	4	>	>	>
5	5	5	≤	≤	≤
6	6	6	GO TO	G T O ß	Ω
7	7	7	GO TO SUB	G S B ß	FOOT-NOTE 2
8	8	8	RETURN	R E T ß	FOOT-NOTE 2
9	9	9	STOP	S T P ß	
0	Ø	Ø	END	E N D ß	FOOT-NOTE 2
.	o	o	JUMP	J M P ß	FOOT-NOTE 2
ENTER EXP	E	↑	IF	I F ß	FOOT-NOTE 2
+	+	+	FLAG IN	F L G ß	
-	-	-	SET / CLEAR FLAG N	S F G ß	FOOT-NOTE 2
*	*	*	SET CLEAR FLAG N / SET CLEAR FLAG N	C F G ß	FOOT-NOTE 2
/	/	/	FIXED N	F X D ß	FOOT-NOTE 2
√‾	.Γ	.Γ	FLOAT N	F L T ß	FOOT-NOTE 2
(	(	(	ENTER	E N T ß	FOOT-NOTE 2
)	)	)	DISPLAY	D S P ß	
;	;	;	PRINT	P R T ß	FOOT-NOTE 2
,	,	,	SPACE N	S P C ß	FOOT-NOTE 2
"	"	FOOT-NOTE 1	NORMAL	N O R ß	FOOT-NOTE 2
			TRACE	T R C ß	FOOT-NOTE 2

KEY	IN QUOTE FIELD? NO	YES	KEY	IN QUOTE FIELD? NO	YES
LOAD	L O D b	FOOT-NOTE 2	s	FOOT-NOTE 3	S
RECORD	R E C b	FOOT-NOTE 2	t	FOOT-NOTE 3	T
A	A	A	u	FOOT-NOTE 3	U
B	B	B	v	FOOT-NOTE 3	V
C	C	C	w	FOOT-NOTE 3	W
X	X	X	s	FOOT-NOTE 3	
Y	Y	Y	%	FOOT-NOTE 3	
Z	Z	Z	&	FOOT-NOTE 3	&
d	FOOT-NOTE 3	D	.	FOOT-NOTE 3	
e	FOOT-NOTE 3	E	?	FOOT-NOTE 3	?
f	FOOT-NOTE 3	F	SPACE	FOOT-NOTE 3	
g	FOOT-NOTE 3	G	These keys have no Mnemonics or Symbols.		
H	FOOT-NOTE 3	H	EXECUTE	—	—
I	FOOT-NOTE 3	I	STORE	—	—
J	FOOT-NOTE 3	J	RUN PROGRAM	—	—
K	FOOT-NOTE 3	K	CLEAR	—	—
L	FOOT-NOTE 3	L	ERASE	—	—
M	FOOT-NOTE 3	M	BACK	—	—
N	FOOT-NOTE 3	N	FORWARD	—	—
O	FOOT-NOTE 3	O	DELETE	—	—
P	FOOT-NOTE 3	P	INSERT	—	—
Q	FOOT-NOTE 3	Q	RECALL	—	—
R	FOOT-NOTE 3	R	LIST	—	—

FOOTNOTES FOR TABLE.

1. The " character never occurs inside a quote field; it is used exclusively to begin or terminate a quote field.
2. This key produces one character with an arbitrary pattern. Sometimes the pattern will vary according to which plug-in ROM's are installed.
3. If this key is used outside of a quote field, NOTE ! ! will result unless the key is defined by a plug-in ROM, in which case the mnemonic or symbol is determined by the ROM.
4. The character β denotes a blank space.

PLUG-IN READ-ONLY MEMORY MODULES

The Mathematics Plug-in Block (henceforth referred to as the Math Block) provides additional mathmatical functions for the Calculator. The keyboard overlay associated with the Math Block is shown in Figure 6. As indicated by this keyboard overlay, the additional mathematical functions provided by the Math Block include logarithms, both natural and common; exponential functions; trigonometric and inverse-trigonometric functions (in degrees, radians, or grads), and others. Use of these functions requires no special programming techniques; once the block is installed, its functions become a part of the calculator, in the same way as, for example, the square root function is part of the calculator.

The Math Block functions are quite straightforward to use; in most cases they require little explanation. The rules and hierarchy applicable to mathematical operations, as described in the calculator's operation, also apply to the operations available with the Math Block. The hierarchy, listed below, is fully discussed in the calculator's operating description.

First: Functions
Exponentiation
Unary Minus
Implied Multiplication
Explicit Multiplication, and
Division
Addition and Subtraction and
Unary Plus

Last: Relational Operators

Some plug-in blocks decrease the amount of programmable memory available to the user, by automatically requiring a portion of that memory for their own internal usage — the Math Block has no such requirement and does not affect memory availability.

The table below described all Math Block Functions. As can be seen from the table, a 'prefix' key (v) is used to redefine certain other keys. Each key which is to be redefined must be immediately preceded by the 'prefix' key — there is no 'continuous prefix' mode.

When the prefix key is pressed, a triangle appears in the display; when the next key is pressed, the lower of the two mnemonics on that key is substituted for the triangle (ASN is displayed, for example, when the sine key is pressed). The triangle cannot now be recovered by pressing the BACK key — in effect the mnemonic for the two keys (the prefix and the next key), once generated, is treated as if it had been generated by one single key.

Reminder: If the argument of a function is negative, then the argument and its sign must be enclosed in parentheses.

SIN (–40) not SIN –40

The functions available with the mathematics plug-in read-only memory module together with syntax information and typical examples are shown in the table below (the left hand side of this table is included on the following page and the right hand side is on the next following page).

Function Type	To Calculate these functions...	...the syntax is:	to obtain the mnemonic.. (b represents blank space)	..press these keys
Logarithms and Inverse Logarithms	natural log x ln x $\log_e$ x (e = 2.71828182848)	LN ⟨quantity⟩ or ((expression))	LN*b* (I)	
	e^x $\log_e^{-1}$ x antilog_e x	EXP ⟨quantity⟩ or ((expression))	EXP*b* (II)	
	common log x $\log_{10}$ x	LOG ⟨quantity⟩ or ((expression))	LOG*b* (III)	
	10^x $\log_{10}^{-1}$ x antilog_{10} x	TN↑ ⟨quantity⟩ or ((expression))	TN↑*b* (IV)	
Circular and Inverse Circular Functions	Select Circular Units	TBL ⟨quantity⟩ quantity = 1 - DEGREES SET 2 - RADIANS SET 3 - GRADS SET	TBL*b* (V)	
	Sine x Sin x	SIN ⟨quantity⟩ or ((expression))	SIN*b* (VI)	
	$\sin^{-1}$ x arc sin x	ASN ⟨quantity⟩ or ((expression))	ASN*b* (VII)	
	Cosine x cos x	COS ⟨quantity⟩ or ((expression))	COS*b* (VIII)	
	$\cos^{-1}$ x arc cos x	ACS ⟨quantity⟩ or ((expression))	ACS*b* (IX)	
	Tangent x tan x	TAN ⟨quantity⟩ or ((expression))	TAN*b* (X)	
	$\tan^{-1}$ x arc tan x	ATN ⟨quantity⟩ or ((expression))	ATN*b* (XI)	
Miscellaneous Functions	Integer x Int x	INT ⟨quantity⟩ or ((expression))	INT*b* (XII)	
	Absolute value of x |x|	ABS ⟨quantity⟩ or ((expression))	ABS*b* (XIII)	
	Exponentiation "raised to the power of"	⟨quantity⟩ or ⟨expression⟩ ↑ ⟨quantity⟩ or ((expression))	↑ (XIV)	
	pi π 3.14159265360	⟨constant⟩	π (XV)	
Initialize Program	Clear Data Storage and Flags	TBL ⟨quantity⟩ quantity = 4 - clear all available R registers 5 - clear all alphabetic registers 6 - clear all flags	TBL*b* (XVI)	

Typical Statements	Range of argument (x)	Remarks
LN 6.2; LN A; LN (X-1); -5LN (4(X-1));	$x > 0$	—
EXP .4; EXP B; EXP (-1); 3((EXP A-EXP (-A))/2)→X;	$-225.65 < x < 227.95$	—
Similar to LN	$x > 0$	—
Similar to EXP	$\|x\| < 10^{99}$	—
TBL 1; TBL 2; TBL 3; TBL Ø; (displays current units)	—	Once set, units remain set until deliberately changed. At turn-on 'DEGREES SET' is automatically assumed.
SIN 3Ø; SIN B; SIN (-45); A 2((1-COS B)↑2+(SIN B) 2)→C;	$\|x\| \leq 1 \times 10^{10}$	
ASN . 7Ø7; ASN (-.3);	$\|x\| \leq 1$	Calculates principle value only: $\theta = \sin^{-1} x; -90^0 \leq \theta \leq +90^0$
Similar to SIN	$\|x\| \leq 1 \times 10^{10}$	
Similar to ASN	$\|x\| \leq 1$	Calculates principle value only: $\theta = \cos^{-1} x; 0^0 \leq \theta \leq +180^0$
Similar to SIN	$\|x\| \leq 1 \times 10^{10}$	
Similar to ASN	$\|x\| < 10^{99}$	Calculates principle value only: $\theta = \tan^{-1} x; -90^0 \leq \theta \leq +90^0$
INT 4.6; INT A; INT (-7.2); INT (A/B+1.Ø3);	—	Eliminates fractional part of value; does not affect sign or integer value.
ABS (-4); ABS A; ABS (A/B+1.Ø3);	—	Sets value positive without otherwise changing the value
5↑2; 5↑(-2); (A+B)↑(X/Y-3);	For A↑B if A<0, then B must be an integer	-2↑4 is equal to -16; (-2)↑4 is equal to +16
2πA; 18ØB/π→C;	—	—
TBL 4; clears all available R-Registers. TBL 5; clears registers A,B,C,X, Y and Z. TBL 6; clears flags 0 through 15.	—	See 'Select Circular Units' for TBL 0, 1, 2, 3. TBL 7, 8, 9 not used

The legends appearing on the keys in the above table are as follows:-

(I) ln / ▼ exp

(II) ▼ , ln / ▼ exp

(III) log / ▼ tn ↑

(IV) ▼ , log / ▼ tn ↑

(V) TABLE N , (n)

(VI) sin / ▼ sin

(VII) ▼ , sin / ▼ asn

(VIII) cos / ▼ acs

(IX) ▼ , cos / ▼ acs

(X) tan / ▼ atn

(XI) ▼ , tan / ▼ atn

(XII) int / ▼ abs

(XIII) ▼ , int / ▼ abs

(XIV) ↑

(XIV) π

(XVI) TABLE N (n)

The calculator's User Definable Functions Accessory includes a 1024-bit plug-in ROM with three 10 key overlays shown in Figures 8A-C. Five keys are used for control and twenty-five keys are available for definition if no other plug-in ROM's are in the machine. Fifteen keys are available if one other plug-in ROM is used and five are available if both other ROM slots are used. The uses of this accessory are described below.

In a program it frequently happens that some basic calculation is needed at several different places. It is clumsy, wasteful and error prone to duplicate the necessary statements each time they are needed. It is easier and more desirable to write them once and refer to the statements as the calculation is required. This capability is provided by subroutines and functions. Here we describe the basic subroutine and function capabilities of the calculator and how they are extended with the USER DEFINABLE FUNCTIONS accessory.

The calculator has basic subroutine capabilities provided by the GO SUB and RETURN keys. These keys allow one or more lines in the main program to be called as a subroutine by jumping to the first line with the GO SUB statement and returning to the main program by executing a RETURN statement. For example, it may be necessary to set the first ten R register to zeros at several places in the program. This job can be accomplished with the following program using a subroutine labeled "ZERO" as follows:

```
0:   GO SUB "ZERO"
      .
      .
20:  GO SUB "ZERO"          Program with three calls to "ZERO"
      .
      .
35:  GO SUb "ZERO"
      .
      .
      .
40: "ZERO"
41: 10 → Z                  }
42: Z – 1 →Z; 0→RZ; IF Z>0; GTO +0   } Subroutine zero
43: RETURN                  }
```

The calls to "ZERO" from lines 0, 20 and 35 cause the ten R registers to be cleared before returning to lines 1, 21, and 36, respectively. The usage of subroutine "ZERO" clearly saves space since the code in lines 41 through 43 need not be duplicated. In addition, as the program is segmented into subroutines it becomes easier to read and understand. If the subroutine is useful to others, it may be incorporated in their programs not only to save space but to save time writing their programs.

In the simple example, subroutine "ZERO" always does *exactly* the same job: setting the first ten R registers to zero. A more general subroutine would have the capability to set any ten consecutive R registers to zeros starting at R(J). To accomplish this, the subroutine must be altered and the value of the *parameter* J must be known by (or passed to) the subroutine. This value could be stored in the X register before calling the subroutine and the program could be changed as follows:

```
0:  0→X; GO SUB "ZERO 1"
     .
     .
20: 40→X; GO SUB "ZERO 1"
     .
     .
35: 30→X; GO SUB "ZERO 1"
     .
     .
40: "ZERO 1"
41: 10 → Z
42: Z – 1 → Z; 0 → R(X+Z), if Z>0; GTO + 0
43: RETURN
```

The subroutine "ZERO 1" clears R registers 0—9, 40—49 and 30—39 in lines 0, 20 and 35, respectively. The programmer must be careful, however, since the subroutine uses both the X and Z registers. These registers must be saved if they contain valued information when the subroutine is called. This bookkeeping complicates using the subroutine and makes it less attractive and more conductive to errors. The problems become even worse as more parameters must be passed to the subroutine and as more working registers, such as Z, must be made available. These problems are circumvented by using advanced features found in the USER DEFINABLE FUNCTION ROM.

In addition the USER DEFINABLE ROM includes the concept of a function. A function differs from a subroutine in that the name of a function has a value associated with it. Therefore, function names can appear in any arithmetic expression to reference the value associated with the functions such as the names A, B, C, X, Y, Z, and R are used for registers. For example,

SIN, COS, LN and EXP

are functions which have values associated with their names and

SIN (LN A) — COS (EXP B) → X

is a valid arithmetic statement containing several functions.

While some standard functions are built in to the calculator, it is desirable to be able to define other functions and have them work in the same manner that the functions sin, cos, ln, exp, etc. work. For example, if a solution of a problem required hyperbolic functions, it would be desirable to define the functions and write statements like

SINH (A + B) – COSH (A – B) → X

The problems encountered in defining functions are similar to those of writing subroutines. Parameters of functions (arguments) must be known by or passed to the function and the working registers must be made available to the function so temporary results may be stored during the calculations. Defining functions differ from defining subroutines in that the value must be assigned to the function. The USER DEFINABLE FUNCTION block provides capabilities to solve these problems.

The option block has key arrangements as shown in Figures 8A-C. Keys FA, FB, FC, FD and FE are assignable to any five subroutines or functions. GA through GJ and HA through HJ are also assignable in the absence of one or two other ROM blocks, thus extending the capacity to 15 or 25 functions or subroutines. The remaining five keys facilitate defining and calling these functions and subroutines.

Subroutines and functions that are defined with the USER DEFINABLE FUNCTION block are similar in structure to the main program: each routine is a list of one or more statements, numbered from zero, followed by an END. To define a simple subroutine to calculate the volume of a sphere and assign this subroutine to the FA key. First press

GTO FA EXECUTE

This places the machine in define subroutine mode related to key FA. Any other assignable key could be used in place of FA. Next, to define the subroutine for calculating the volume, STORE

0: 4/3*πZZZ→Z

1: END

Storing the END returns the machine to the normal mode of operation. To use this subroutine to calculate the volume of a sphere with radius 5, press

5: FA EXECUTE

which is equivalent to

5→Z; GSB FA EXECUTE

The Z register is displayed. To call the subroutine from a program STORE

3: 5→Z; GSB FA

The five control keys (left keys of Figure 8A) extend these basic subroutine capabilities to include immediate execute as well as parameter passing and function subprograms. These keys are described below.

IMMEDIATE EXECUTE. The immediate execute key (displayed as IEX) is used in making the calculator respond immediately to the depression of a key without pressing EXECUTE. The IEX must be the first statement of the subroutine for the key to respond in this manner. When the key associated with such a subroutine is depressed, the routine is executed without pressing EXECUTE. For example, if the previous program was changed to

0: IEX

1: 4/3*πZZZ Z

2: END

then merely press

5 FA

to invoke the routine. This specialized execution mode is valuable in simplifying keyboard operation to increase efficiency and productivity when moving similar calculations must be made from the keyboard.

CALL. To call a subroutine with parameters the CLL must be used. This key is used to indicate that a list of parameters will follow the subroutine name. Otherwise, the key is used exactly as GSB key. That is,

GSB FA (no parameters)

CLL FA (A, 5, B+X) (parameters)

The CLL statement should be the last statement of a line. The parameters need not be enclosed in parentheses.

PARAMETER. The P( ) or parameter key is used to access parameters that are being passed to subroutines and functions and is probably the most heavily used key of this ROM block. In addition to accessing parameters, the P( ) key may be used to create and access memory that is used temporarily as working registers while the subroutine is being executed. Accessing parameters and working registers is done with the P( ) key without affecting the A, B, C, X, Y, Z, or R registers.

The P( ) key is used exactly like the R( ) key but it references a sequence of parameters registers instead of the R registers. For example, if a subroutine FB is called with three parameters, P1 references the first parameter, P2 references the second, etc. This is,

```
CLL FB (A,  5,  X—B)
        ↕   ↓   ↓
        P1  P2  P3
```

In this CLL, P1 references the A register, P2 and P3 reference memory locations where 5 and the value of X—B are stored temporarily during the execution of subroutine FB. The calculation of X—B is made and placed in a temporary location each time the CLL statement is executed before executing subroutine FB.

Temporary working registers may be created and accessed by using the P( ) key with subscripts higher in value than the number of parameters being passed. For example, subroutine FB has three parameters (P1, P2, P3). P4, P5, etc. could be used as working registers. Obviously, the number of such registers is limited

since the calculator will run out of internal temporary storage eventually. An exact limit cannot be given since it is dependent on the availability of memory when the subroutine is initiated.

As the first example, consider rewriting subroutine "ZERO 1" to zero the specified ten R registers without destroying the value of the X or Z registers as the previous routine did. One parameter P1 must be passed replacing X and one working register P2 is used in place of Z. The necessary statements follow.

```
PRESS   GTO FA EXECUTE
STORE   0:   "ZERO 1"
        1:   10→P2
        2:   P2—1→P2; 0→R(01+P2); IF P2>0; GTO+0
        3:   END
```

Then, CLL of the form

```
40→X; GO SUB "ZERO 1"
```

are replaced pressing

```
CLL FA 4 0 STORE
```

which is displayed as

```
20:  CLL ZERO 1 40
```

since the subroutine is started with the label "ZERO 1". The new routine operates as prescribed without destroying the values of either register X or Z freeing them for other purposes.

Another example is a routine to increment a register. The one parameter of this subroutine specifies the register to be incremented:

```
PRESS       GTO FC EXECUTE
STORE   0:  "INCR"
        1:  P1+1→P1
        2:  END
```

INCR may be called by

```
10:  CLL INCR A
```

to increment the A register or

```
20:  CLL INCR R (A+B)
```

to increment the R register specified by A+B. This example shows that a parameter may be used to return a result as well as access a value. Any number of parameters may be used in calling a subroutine.

DEFINE. A function differs from a subroutine in that it has a value associated with its name and, therefore, can be part of an expression. The DEF/→F key allows functions to be defined in the calculator. The key has two uses as its label indicates. First, it is used to place the machine in function definition mode DEF. Secondly, once the calculator is in function definition mode, it is used to assign a value to the function →F.

To place the calculator in *function definition* mode,

```
PRESS       DEF FA EXECUTE
```

This is analogous to placing the machine in subroutine definition mode; that is pressing

GTO FA EXECUTE.

after placing the machine in function definition mode, the function is defined exactly as a subroutine with parameters except the →F allows a value to be assigned to the function.

As an example consider writing a function to define the hyperbolic sin function.

$$\text{Sinh } X = \frac{e^{x} - e^{-x}}{2}$$

as the FD key. First, to place the calculator in function definition mode,

PRESS DEF FD EXECUTE

to define the sinh function,

STORE 0: "SINH"

1: (EXP P1 — EXP (–P1))/2→F

2: END

to use the function, the FD key is referenced just like the SIN key. For example,

PRESS FD (5) + FD (4) EXECUTE

which is displayed as

SINH (5) + SINH (4)

before EXECUTE is pressed since the definition begins with the label "SINH". Similarly,

5: SINH (A+B) / SINH (A–B) → A

can be stored as a program line. The machine truly behaves as if it has a "built in" capability to calculate hyperbolic sines.

As a second example, the maximum value function is programmed. This function has two parameters and is assigned the value of the larger of the two parameters. First,

PRESS DEF FE EXECUTE

and STORE 0: "MAX"

1: P1 → F

2: IF P2>P1; P2→F

3: END

Notice that P1 is assumed to be the larger of the two parameters in line 1, and line 2 makes a correction if this is not the case. This function can be used to calculate and store the product of two maximum values as follows:

MAX (6, 9) MAX (–5, –4) → RA

or MAX (AB — C, 5) MAX (Z↑3, 5—A) → RC

Performing similar operations without using this function capability would require

several registers to store intermediate results and would be very had to read and understand in comparison.

SCRATCH. The SCR is used for several function. Its primary use is to delete a user defined subroutine or function from memory to allow a key to be used for other programs or to increase the amount of memory available for the main program. To delete function FA,

PRESS SCR FA EXECUTE

to delete two (or more),

PRESS SCR FB, FC EXECUTE

special functions of this key included recording and loading of programs. To rcord all programs in memory in the order stored,

PRESS GTO SCR: REC EXECUTE

to load these programs,

PRESS GTO SCR: LOD EXECUTE

to record one function or subroutine per one half card, place one subroutine or function in the machine and

PRESS GTO FA (OR OTHER KEY DEFINED): REC EXECUTE

similarly, to configure a machine from a library created in this manner, order the functions and subroutines and PRESS

GTO FA: LOD EXECUTE

GTO FB: LOD EXECUTE

ETC.

To list function FA,

press GTO FA LIST

The ability to configure the calculator in this manner makes it possible to customize the calculator from one problem to the next without reprogramming, entering and debugging the functions and subroutines needed. This ability combined with the capability of the calculator to modify the keyboard with a variety of plug-in ROMS allows versatility never before found in a calculator.

In summary, the USER DEFINABLE FUNCTIONS ROM for the calculator greatly extends the capabilities of the calculator. It has been shown how the block is used to write general purpose subroutines and functions. These routines communicate with the main program by parameter passing and allow working registers within the subroutine to be established and accessed. These features allow the user to define routines that do not require or destroy the content of the A, B, C, X, Y, Z and R registers. Therefore, the programmer is relieved of all the bookkeeping that is associated with calling a subroutine when parameters must be placed in specified registers; these registers usually have to be saved before storing parameters and restored after calling the subroutine. The programs written with this required bookkeeping becomes clumsy, obscure, hard to debug, and in general discourages the use of subroutines and functions.

Another advantage of the USER DEFINABLE FUNCTIONS ROM is its ability to define functions (subroutines that have a value associated with their names such as SIN and LN) that exactly imitate the behavior of the built in functions of the calculator. This allows the capabilities of the machine to be extended naturally when a problem that is based on different functions is encountered.

The option block also allows a library of general purpose subroutines and functions to be established and used easily. This ablility greatly emancipates the

programmer by allowing him to borrow something written by another with a minimum of effort.

In general, the USER DEFINABLE FUNCTIONS extend the capabilities of the calculator to make the machine easier and more natural to program. It may be the user's most valuable addition to the calculator.

The Peripheral Control I accessory includes a 1024 bit addition to the read-only memory and a keyboard overlay, as shown in Figure 7, to identify the ten keys associated with this accessory. Of these ten keys five are for the operation of a X—Y plotter, two for typewriter and the remaining three for general control of both input and output peripherals. A description of these 10 keys follow. First the five plotter keys.

SCALE. This key allows the user to scale the physical limits of the X—Y plotter to any units he desires. The syntax for this key is as follows:

$$\text{SCL } X_{min}, X_{max}, Y_{min}, Y_{max}$$

The four limits after the SCL follow the rules of any parameter list in the calculator. After SCL is executed all communication with the X—Y plotter may be in the ranges specified by the scale statement. An example: if the user were going to plot Y=8sin X for -2 X 2 and with a range for Y of -10 Y 10 he would specify:

$$\text{SCL } -2\pi, 2\pi, -10, 10$$

(this same example will be used in other key descriptions).

AXES. This key allows the user to draw both X and Y axes on his plot with one instruction. The syntax is as follows:

$$\text{AXE } X_o, Y_o, \Delta X, \Delta Y$$

Here X_o, Y_o specifys the values on the scaled plot in user units where the axes will cross. The optional X and Y specify the distance between tic marks on the X and Y axes respectively. These tic marks will be generated starting from the origin. For the example cited previously:

$$\text{AXE } 0, 0, \pi/4, .1$$

PEN UP. This instruction is used without parameters and instructs the X—Y plotter to raise the marking pen off of the paper staying in the same X, Y location.

LETTER. This instruction with its parameter list sets up the X—Y plotter for plotting alphameric characters. The syntax follows:

$$\text{LTR X, Y, hwe}$$

Here X, Y specifies the X, Y location, in user units, of the lower left hand corner of the first alphameric character to be plotted. hwe is a three digit number that specifies the size of the letters and the direction of lettering on the plot. The h is a digit in the range $1 \leq h \leq 9$ which specifies the alphameric character height. The actual height is .64h percent of full scale (i.e. $Y_{max} - Y_{min}$ if plotted horizontally). Similarly w is a digit in the range $1 \leq w \leq 9$ which specifies the alphameric character width. The actual width is .64w percent of full scale (i.e. $X_{max} - X_{min}$ if plotter horizontally). The e must fall in the range $1 \leq e \leq 4$. 1 allows plotting horizontally left to right, 2 is vertically bottom to top (right reading), 3 horizontally right to left (up-side-down) and 4 is vertically top to bottom (left reading). To set up a lable for the example specified above:

$$\text{LTR } \pi/2, 8, 321$$

the actual plotting of the alphameric characters is accomplished with the plot key described below.

PLOT. Plot does three things, allows point plotting, alphameric label plotting and numeric results plotting.

$$\text{PLT X, Y}$$

The above syntax is used for point plotting. It must have two parameters which

have the same limitations as other calculator parameters. If the X—Y plotter pen is up on execution of this statement it will remain up, the pen will travel to the X, Y location specified then the pen will go down to the paper. If the pen is down on execution of the statement it will remain down as it travels to the X, Y location, thereby drawing a line from the old location to the one specified. To plot the sin curve in our previous example assuming the register X is our variable X then we could scale, draw axes and plot with a program shown below.

```
0: TBL 2                    (set calculator to radians)
1: SCL -2π, 2π, -10, 10
2: AXE 0, 0, π/4, .1
3: -2π → X
4: PLT X, 10 sin X
5: .1 + X→X
6: IF X≤ 2π; GTO 4
```

Plot is used to plot alphameric labels with a single parameter enclosed in quotes.

PLT "(Alphameric Label)"

To continue our program from above and plot a heading on it would require a letter statement and a plot statement.

```
7: LTR π/2, 8, 321
8: PLT "SIN X VS. X"
```

The final syntax for plot is:

PLT (numeric or arithmetic expression)

Where again we only have one parameter. On execution this will plot the numeric or value of the arithmetic expression in the format specified by the FIXED N or FLOAT N keys on the calculator. To extend our example program to plot a label for the left most X axes tic mark.

```
9: FXD 3
10: LTR -2π, -1, 211
11: PLT -2π
```

This would give at the location -2π, -1:

−6.283

This concludes the description of the five plotter keys. Next the two typewriter keys.

FORMAT. This key allows specification of a format for alphameric character output. It is of the form:

fmt Spec 1, Spec 2 . . . Spec n

Where Spec 1 etc. are either conversion specifications for converting the internal calculator floating point numbers to a desired output form or edit specifications which allow location manipulation, alphameric outputs and control of special typewriter commands.

Conversion specifications have three forms:

r FLT w.d Floating point number (w≥ d+7)

r FXD w.d Fixed point number

r FXD w.o Integer without decimal point

In these cases r specifies the number of times the conversion specification will be repeated. FLT or FXD comes from depression of the FLOAT N or FIXED N keys on the calculator. The w specifies the total field width for this output and d the number of digits to the right of the decimal point. r, w and d must be integer numerics.

Editing specifications may include the following:

nX

Which specifies a blank field of n characters. n must be an integer, X is the X key on the calculator.

r/

Specifies r carriage return, line feeds to the typewriter. r must be an integer, / is the / (divide) key on the calculator keyboard.

Z

Normally a carriage return, line feed is given automatically at the end of each FMT statement. If the calculator key Z is included as a specification in the FMT the automatic carriage return, line feed is suppressed.

r "(alphameric label & typewriter control)"

In this r again is an integer specifying the number of times this specification will be repeated. Inside the quotes may be used a variety of calculator keys to allow outputting to the typewriter the following:

Upper case alphabet

Lower case alphabet

Special symbols

$ % & ' ? space / * – + . ,

( ) _ : ~ > < = ≠ ! @

Special controls

Red ribbon, black ribbon, back space, tab, line feed, carriage return, clear all tabs, set tab

The key assignments for these characters and controls for the calculator keyboard are shown in Figure 11. To accommodate all of these symbols on the limited number of keys it is necessary to use the shift key. This is the normal DISPLAY key. When used in a quote field it gives the special symbol Γ in the display. The shift key is a toggle type. The entry to a quote field always sets the keyboard to upper case (unshifted) then on each encounter with Γ after that the keyboard mode is toggled between upper case (unshifted) and lower case (shifted). For example, to output Bill Hewlett on the typewriter would require a program line as follows:

FMT "BΓILLΓ HΓEWLETT"; TYP

TYPE. The FMT sets up the actual output specification but does not specify the actual output. To do this requires the TYPE key. This is followed by a

parameter list where parameters are numerics, registers or expressions. On execution of the TYP first, the last encountered FMT is found. The FMT specifications are sequentially scanned left to right, starting at extreme left, outputting edit specifications to the typewriter until the 1st conversion specification (or all specifications executed) is found. Then the first parameter in the TYP statement is evaluated and this is outputted with first conversion specification of the FMT. Scanning of FMT specifications then continues, outputting edit specifications until the next conversion specifications is found. Then again we go to the TYP parameter list and evaluate the second parameter and output to the typewriter with this conversion. This process continues until either the end of FMT specifications or the end of the TYP parameters is reached.

If the end of the FMT specifications is reached before the end of TYP a carriage return, line feed is given and the process described above continues starting with the beginning of the FMT specifications again.

If the end of FMT is reached at the same time the TYP parameter list is depleted then carriage return, line feed is given and the calculator program continues with the next statement.

If after finding a conversion specification in the FMT and there is no corresponding TYP parameter (end of parameter list) a carriage return, line feed is given and execution of the program continues with the next statement. In this case the next TYP will start again with the first FMT specification, not where it discontinued previously.

Any time a conversion specification cannot accommodate the number to be outputted first, an attempt will be made to go to floating point (maintaining the same w.d) if it still overflows the field will be filled with dollar signs.

The following example will illustrate the TYP with FMT. To output a trig table as shown below:

TRIGONOMETRIC TABLE

DEGREES	SIN	COS	TAN
0	0.000	1.000	0.000
1	.018	1.000	.016
2	.035	.999	.035
.			
.			
.			
89	1.000	.018	57.29
90	1.000	0.000	$$$$$$

Use a program as shown:

```
0: SFG 14; TBL 1
1: FMT 5X, "TRIGONOMETRIC TABLE",/
2: TYP
3: FMT "DEGREES SIN COS TAN",/
4: TYP
5: O→X
6: FMT FXD 6.0, 2 FXD 8.3, 2X, FXD 7.3,/
7: TYP SIN X, COS X, TAN X
```

```
8:  1 + X → X
9:  IF X≤ 90; GTO 7
10: STP
```

There is a default option with the TYP statement if no FMT is specified. This default has four features.

The first is, the setting specified by the calculator FIXED N or FLOAT N keys is assumed for outputting data.

The second, four fields of 18 characters each are assumed for field width specifications across the page.

The third, the parameter list of the TYP statement may now include label parameters, i.e. "(Alphameric Label)". These will be outputted in the 18 character fields. These labels may only include the unshifted (upper case) characters. They will use as many 18 character fields as necessary.

The final feature is, all outputs are right justified in the 18 character fields. The following examples will illustrate:

```
TYP "HEADING"
```

```
HEADING
```

```
1→A; 2→B; 3→C; FXD 0
TYP A; B, C
    1              2              3
```

```
FXD 2
TYP "X=", 1.345
X=            1.35
```

The final 3 keys of this accessory are described now. These are for general peripheral control. They use the concept of select codes. The select code is a number assigned to a particular peripheral to identify it. A digitizor may be select code 3, a paper tape punch select code 13, etc. Select code will be designated by SC. Usually this can be a number, register or arithmetic expression. When evaluated the integer value will be used.

WRITE. This key is the output key and like TYP is used with the FMT key. It does not have the default option as the TYP. It's description is identical to the TYP key with these exceptions. In this case the peripheral to be outputted to must be identified by SC. This must be the first parameter in the WRT parameter list. The actual outputs out of the calculator are ASCII codes. How they are interpreted is determined by the peripheral itself. For example, the teletype interpretation of the keyboard keys is shown in Figure 12.

READ. This allows data input from external peripherals. The syntax is:

```
RED SC, A, B R12
```

The first parameter must be the select code of the inputting peripheral. The following parameters must be register names that specify registers to accept the inputted data. On execution the calculator will send instructions to turn on the peripheral device and then standby for ASCII characters coming in. It will accept the characters 0—9 . , / – + E (enter exponent) and any other characters except leading spaces and one space following the enter exponent will be interpreted as delimiters and cause the number inputted to be stored in the register specified. The sequence is repeated for each register in the parameter list.

TRANSFER. This key is used to transfer directly from one external peripheral to another.

```
TFR  SC1,  SC2
```

This does a direct transfer from the peripheral with SC_1 to the peripheral with SC_2 (e.g. a paper tape reader to a type-writer). The transfer is initiated when the TFR is executed and terminates on transferring a 8 bit code of 00 000 011.

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 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

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>.

REGISTER REFERENCE OF A OR B REGISTER: If the location <A> or <B> 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 address 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 <n> which permits a shift of one through 16 bits; that is, $1 \leq n \leq 16$. If <n> 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.

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.

ABR n Arithmetic right shift of B. Otherwise identical to AAR.

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.

SAL n Shift A left. The A register is shifted left n places with the n least significant bits equal to zero.

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.

RBR n Rotate B right. Otherwise identical to RAR.

Alter-Skip Group

The sixteen alter-skip instructions all contain a 5-bit variable skip field <n> 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 <n> is the number to be added to the current address, (skip in *forward* direction), and the number <−n> is the number to be subtracted from the

current address, (skip in negative direction). If <n> is omitted, it will be interpreted as a ONE.

<n> = 0	CODE=00000	REPEAT SAME INSTRUCTION
<n> = 1	CODE=00001	DO NEXT INSTRUCTION
<n> = 2	CODE=00010	SKIP ONE INSTRUCTION
<n> = 15	CODE=01111	ADD 15 TO ADDRESS
<n> = −1	CODE=11111	DO PREVIOUS INSTRUCTION
<n> = −16	CODE=10000	SUBTRACT 16 FROM ADDRESS
<n> = nothing	CODE=00001	DO NEXT INSTRUCTION

The alter bits consist of bits 10 and bits 4. The letter <ζ> following the instruction places a ONE in bit 10 which causes the tested bit to be *set* after the test. Similarly the letter <ε> 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.

RZA n Skip if A not zero. This is a "Reverse Sense" skip of SZA.

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.

SIB n Skip if B zero; then increment B. Otherwise identical to SIA.

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.

SIA 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.

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.

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.

SES n,S/C Skip if Least Significant bit of E is Set. If bit 0 of the E register in 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-Esecute-DMA Group

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.

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.

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.

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

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.

Input/Output Group (IOG)

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.

STF <SC> Set the flag. Set the flag flip-flop of the channel indicated by select code <SC>.

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.

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.

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.

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.

OT* <SC> H/C Outpout 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 flop after execution of OT*. The different select codes allow different functions to take place after loading the I/O register.

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.

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.

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.

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.

LI* <OI> 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.

LI* <OO> The least significant 8 bits of the I/O register are loaded into the most significant locations in the A or B register.

MI* <OI> 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 M1* has been executed.

MI* <OO> 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: $<A_{0-3}>$ means: contents of A-register bit 0 to 3

AR 1 is a mnemonix for arithmetic pseudo-register located in R/W memory on addresses 1744 to 1747 (octal)

AR 2 is a mnemonix for arithmetic pseudo-register located in R/W memory on addresses 1754 to 1757 (octal).

D_i means: mantissas 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>, <B>, <E> unchanged.

MOV Move overflow
The contents of E-register is transferred to A_{0-3}. Rest of A-register and E-register are filled by zeros.
<B> unchanged.

CLR Clear a floating-point data register in R/W memory on location <A>
ZERO → <A>, <A> +1, <A> +2, <A> +3
<A>, <B>, <E> unchanged.

XFR Floating-point data transfer in R/W memory from location <A> to location <B>.
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.

$<B_{0-3}> = n$ (binary coded)

1st shift: $<A_{0-3}> \rightarrow D_1$; $D_i \rightarrow D_{i+1}$; D_{12} is lost

jth shift: $\Theta \rightarrow D_1$; $D_i \rightarrow D_{i+1}$; D_{12} is lost

nth shift: $\Theta \rightarrow D_1$; $D_i \rightarrow D_{i+1}$; $D_{12} \rightarrow A_{0-3}$

$\Theta \rightarrow E$, A_{4-15}

each shift: $<B_{0-3}> - 1 \rightarrow B_{0-3}$

$<B_{4-15}>$ 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.

$\Theta \rightarrow D_{12}$; $D_i \rightarrow D_{i-1}$; $D_1 \rightarrow A_{0-3}$

<B> unchanged

DRS AR1 mantissa is shifted to right once.
Exponent word remains unchanged.

$\Theta \rightarrow D_1$; $D_i \rightarrow D_{i+1}$; $D_{12} \rightarrow A_{0-3}$

ZERO → E and A_{4-15}

<B> unchanged

DLS AR1 mantissa is shifted to left once.
Exponent word remains unchanged.

$<A_{0-3}> \rightarrow D_{12}$; $D_i \rightarrow D_{i-1}$; $D_1 \rightarrow A_{0-3}$

$\Theta \rightarrow E$, A_{4-15}

<B> unchanged.

FXA Fixed-point addition
Mantissa 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.

<AR2> + <AR1> + DC → AR2

DC = if <E> was 0000 before routine execution

DC = 1 if <E> was 1111 before routine execution

<B>, <AR1> unchanged.

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.

<AR2> + <AR1> *$<B_{0-3}>$ + DC → AR2

DC = 0 if <E> was 0000 before routine execution
DC = 1 if <E> was 1111 before routine execution
ZERO → E, A_{4-15}

<AR1> unchanged.

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 <B>.

1st addition: <AR2> + <AR1> + DC → AR2

DC = 0 if <E> was 0000 before routine execution
DC = 1 if <E> was 1111 before routine execution
next additions: <AR2> + <AR1> AR2
ZERO → E
<AR1> unchanged.

CMX 10's complement of AR1 mantissa is placed back to AR1, and ZERO is set into E-register. Exponent words remains unchanged.
<B> unchanged.

CMY 10's complement of AR2 mantissa.
Otherwise identical to CMY.

MDI Mantissa decimal increment.
Mantissa on location <A> is incremented by decimal ONE one D_{12} level, result is placed back into the same location, and zero is set into E-reg.
Exponent word is unchanged.
When overflow occurs, result mantissa will be
1,000 0000 0000 (dec)
and 0001 (bin) will be set into E-reg.
<B> unchanged.

NMR 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 ($<B_{4-15}> = 0$)

when $<B_{0-3}> = 0,1,2,\ldots, 11$ (dec) ⇛ <E> = 0000

when $<B_{0-3}> = 12$ (dec) ⇛ mantissa is zero, and <E> = 0001.

Exponent word remains unchanged
<A> 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*	D/I	0 0 0	A/B Z/C	← MEMORY ADDRESS →
	-1----	CP*	D/I	0 0 1	A/B Z/C	
	-2----	LD*	D/I	0 1 0	A/B Z/C	
	-3----	ST*	D/I	0 1 1	A/B Z/C	
	-4----	IOR	D/I	1 0 0	0 Z/C	
	-4----	ISZ	D/I	1 0 0	1 Z/C	
	-5----	AND	D/I	1 0 1	0 Z/C	
	-5----	DSZ	D/I	1 0 1	1 Z/C	
	-6----	JSM	D/I	1 1 0	1 Z/C	
	-6----	JMP	D/I	1 1 0	1 Z/C	
SHIFT ROTATE GROUP	07---0	A*R	0	1 1 1	A/B - -	SHIFT CODE - 0 0 0 0
	07---2	S*R	0	1 1 1	A/B - -	- 0 0 1 0
	07---4	S*L	0	1 1 1	A/B - -	- 0 1 0 0
	07---6	R*R	0	1 1 1	A/B - -	- 0 1 1 0
ALTER SKIP GROUP	07---0	SZ*	0	1 1 1	A/B 0	← SKIP CODE → 0 1 0 0 0
	07---0	RZ*	0	1 1 1	A/B 1	0 1 0 0 0
	07---0	RI*	0	1 1 1	A/B 0	1 1 0 0 0
	07---0	RI*	0	1 1 1	A/B 1	1 1 0 0 0
	07---1	SL*	0	1 1 1	A/B H/C	H/C 1 0 0 1
	07---2	S*M	0	1 1 1	A/B H/S	H/C 1 0 1 0
	07---3	S*P	0	1 1 1	A/B H/S	H/C 1 0 1 1
	07---4	SES	0	1 1 1	D/I H/C	H/C 1 1 0 0
	07---5	SEC	0	1 1 1	A/B H/C	H/C 1 1 0 1
REGISTER REFERENCE GROUP	07--17	ADA	0	1 1 1	A/B - -	D/I 0 0 0 0 1 1 1
	07--37	ADB	0	1 1 1	A/B - -	D/I 0 0 0 1 1 1 1
	07--57	CPA	0	1 1 1	A/B - -	D/I 0 0 1 0 1 1 1
	07--77	CPB	0	1 1 1	A/B - -	D/I 0 0 1 1 1 1 1
	07--17	LDA	0	1 1 1	A/B - -	D/I 0 1 0 0 1 1 1
	07--37	LDB	0	1 1 1	A/B - -	D/I 0 1 0 1 1 1 1
	07-557	STA	0	1 1 1	A/B - -	1 0 1 1 0 1 1 1 1
	07-577	STB	0	1 1 1	A/B - -	1 0 1 1 1 1 1 1 1
	07--17	IOR	0	1 1 1	A/B - -	D/I 1 0 0 0 1 1 1
	07-637	ISZ	0	1 1 1	A/B - -	1 1 0 0 1 1 1 1 1
	07--57	AND	0	1 1 1	A/B - -	D/I 1 0 1 0 1 1 1
	07-677	DSZ	0	1 1 1	A/B - -	1 1 0 1 1 1 1 1 1
	07-717	JSM	0	1 1 1	A/B - -	1 1 1 0 0 1 1 1 1
	07--37	JMP	0	1 1 1	A/B - -	D/I 1 1 0 1 1 1 1

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	⅛ - -	- - -	0 0 1	1 1 0
	070036	DMA	0 1 1 1	0 - -	- - -	0 1 1	1 1 0
	07-056	CM*	0 1 1 1	⅛ - -	- - -	1 0 1	1 1 0
	07-076	TC*	0 1 1 1	⅛ - -	- - -	1 1 1	1 1 0
INPUT OUTPUT GROUP	1727--	STF	1 1 1 1	- 1 0	1 1 1 1	SELECT CODE	
	1737--	CLF	1 1 1 1	- 1 1	1 1 1 1		
	17-7--	SFC	1 1 1 1	- 1 ⅛	1 1 1 0		
	17-5--	SFS	1 1 1 1	- 1 ⅛	1 0 1 0		
	17-5--	CLC	1 1 1 1	- 1 ⅛	1 0 1 1		
	17-6--	STC	1 1 1 1	- 1 ⅛	1 1 0 0		
	17-1--	OT*	1 1 1 1	⅛ 1 ⅛	0 0 1 1		
	17-2--	LT*	1 1 1 1	⅛ 1 ⅛	0 1 0 1		
	17-0--	MI*	1 1 1 1	⅛ 1 ⅛	0 0 0 1		
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	CLR	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	1 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 1	1 0 0	0 0 0	0 0 0
	170410	DRS	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	FXA	1 1 1 1	0 0 0	1 0 1	1 1 0	0 0 0
	171460	FMP	1 1 1 1	0 0 1	1 0 0	1 1 0	0 0 0
	170420	FDV	1 1 1 1	0 0 0	1 0 0	0 1 0	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	MDI	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 routines and subroutines of basic instructions employed by the calculator and of 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 of each different group of routines, subroutines, and constants is numbered at the top left-hand corner of the page, and 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 page. As noted above, 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".

BASE PAGE 205B—554B 1400B—1777B

```
0001          ASMB,A,L
0003*
0004    00276 ORG 276B  ROUTINE LINKAGES AND SYSTEMS CONSTANTS
0005*
0006* WITH THE EXCEPTION OF 16 WORDS THE
0007* SYMBOL TRANSFORM TABLE FOR THE
0008* 5X7 CHARACTER GENERATION FOR THE PRINTER
0009* AND THE DISPLAY USES:
0010*
0011*    205—262    46 WORDS
0012*
0013*  206    .DSP1 DEF  3000B      (SEE I/O SUP LINE 133)
0014*  207    .DBUF DEF  1660B      (SEE I/O SUP LINE 134)
0015*
0016*  211    .TAB1 DEF  101564B    (SEE I/O SUP LINE 136)
0017*  212    .TAB3 DEF  000203B    (SEE I/O SUP LINE 137)
0018*
0019*  225    .LOD  DEF  3171B      (SEE I/O SUP LINE 140)
0020*  226    .REC  DEF  3043B      (SEE I/O SUP LINE 141)
0021*
0022*  245    .TAB5 DEF  3470B      (SEE I/O SUP LINE 144)
0023*  246    .TAB6 DEF  2100B      (SEE I/O SUP LINE 145)
0024*  247    .TAB7 DEF  1626B      (SEE I/O SUP LINE 146)
0025*  250    .PRT1 DEF  2536B      (SEE I/O SUP LINE 147)
0026*  251    .DSP  DEF  2715B      (SEE I/O SUP LINE 148)
0027*  252    .BF31 DEF  2002B      (SEE I/O SUP LINE 149)
0028*  253    .BF5A DEF  2051B      (SEE I/O SUP LINE 150)
0029*  254    SKCD  DEF  7017B      (SEE I/O SUP LINE 151)
0030*  255    L18A  DEF  2227B      (SEE I/O SUP LINE 152)
0031*  256    .TAB4 DEF  000430B    (SEE I/O SUP LINE 153)
0032*
0033* LINKAGE TO THE BASIC ARITHMETIC
0034* OPERATORS USES:
0035*
0036*    263—275
0037*
0038* THE NEXT AREA 276—554    175 WORDS
0039* IS USED BY THE SYSTEM FOR COMMON CONSTANTS
0040* AND NEEDED LINKAGES. THE AREA IN WHICH THE
0041* CONSTANT IS USED IS INDICATED BY THE FOLLOWING
0042* KEY:
0043*
0044*  COLUMN 1    BASIC MATH AREA 'MATH 1'
0045*         2    I/O SUPERVISOR
0046*         3    MAG CARD LOAD AND RECORD
0047*         4    COMPILER/INTERPRETER
0048*         5    CONTROL SUPERVISOR
0049*         6    UNCOMPILER
0050*
0051  00276  020060  K1     OCT  20060    2
0052  00277  002270  .FMT   DEF  2270B    2
0053  00300  002525  .BXFR  DEF  2525B    2
0054  00301  003271  SYNTX  DEF  3271B    2
0055  00302  003273  EXEQX  DEF  3273B    2
0056  00303  003365  OUT    DEF  3365B    3
0057  00304  002241  .FLT   DEF  2241B    2
```

BASE PAGE 205B—554B 1400B—1777B

0058	00305	002251	.FXD	DEF	2251B	2
0059	00306	002022	.BF5G	DEF	2022B	2
0060	00307	006226	KY6	DEF	6226B	5
0061	00310	002260	.FMT2	DEF	2260B	2
0062	00311	016045	UNCPL	DEF	16045B	6
0063	00312	006723	ENT2	DEF	6723B	5
0064	00313	004055	CMSET	DEF	4055B	4
0065	00314	004071	CMPLR	DEF	4071B	4
0066	00315	004547	INTRP	DEF	4547B	4
0067	00316	005047	XEOL	DEF	5047B	4
0068	00317	007312	.CSIN	DEF	7312B	5
0069	00320	007712	RGTO	DEF	7712B	5
0070	00321	007346	END	DEF	7346B	5
0071	00322	016303	TAB1	DEF	16303B	5
0072	00323	016006	SMON	DEF	16006B	5
0073	00324	002470	.RND1	DEF	2470B	4
0074	00325	102470	CKS1	OCT	102470	CHECKSUM
0075	00326	005616	RECY	DEF	5616B	4
0076	00327	002400	K2	OCT	2400	5
0077	00330	177762	M14	DEC	−14	12
0078	00331	177401	K3	OCT	177401	1
0079	00332	001750	K4	DEF	1750B	1
0080	00333	000521	K5	OCT	521	4
0081	00334	047100	K6	OCT	47100	1
0082	00335	072000	K7	OCT	72000	1
0083	00336	061400	K8	OCT	61400	1
0084	00337	013732	RA	DEF	6106	1
0085	00340	060141	K9	OCT	60141	2
0086	00341	000400	P256	DEC	256	12
0087	00342	000377	P255	DEC	255	15
0088	00343	000220	P144	DEC	144	2
0089	00344	000200	P128	DEC	128	2
0090	00345	000177	P127	DEC	127	2456
0091	00346	004570	XJ	DEF	4570B	4
0092	00347	000120	P80	DEC	80	25
0093	00350	000100	P64	DEC	64	25
0094	00351	000077	P63	DEC	63	156
0095	00352	000021	P17	DEC	17	56
0096	00353	000064	P52	DEC	52	46
0097	00354	007436	UNC1	DEF	7436B	5
0098	00355	000060	P48	DEC	48	245
0099	00356	000057	P47	DEC	47	34
0100	00357	007417	ISTOR	DEF	7417B	5
0101	00360	000040	P32	DEC	32	24
0102	00361	000037	P31	DEC	31	235
0103	00362	004560	XRH	DEF	4560B	4
0104	00363	171464	K10	OCT	171464	4
0105	00364	000020	P16	DEC	16	245
0106	00365	000017	P15	DEC	15	24
0107	00366	000016	P14	DEC	14	246
0108	00367	000015	P13	DEC	13	2456
0109	00370	000014	P12	DEC	12	245
0110	00371	000013	P11	DEC	11	46
0111	00372	000012	P10	DEC	10	146
0112	00373	000011	P9	DEC	9	4
0113	00374	000010	P8	DEC	8	145

BASE PAGE 205B—554B 1400B—1777B

0114	00375	000007	P7	DEC	7	2345
0115	00376	000006	P6	DEC	6	2345
0116	00377	000005	P5	DEC	5	1245
0117	00400	000004	P4	DEC	4	12345
0118	00401	000003	P3	DEC	3	123456
0119	00402	000002	P2	DEC	2	123456
0120	00403	000001	P1	DEC	1	123456
0121	00404	000105	K11	OCT	105	2
0122	00405	177777	M1	DEC	–1	123456
0123	00406	177776	M2	DEC	–2	245
0124	00407	177775	M3	DEC	–3	1245
0125	00410	177774	M4	DEC	–4	1245
0126	00411	177773	M5	DEC	–5	125
0127	00412	177772	M6	DEC	–6	3
0128	00413	177771	M7	DEC	–7	24
0129	00414	001570	AISP	DEF	1570B	45
0130	00415	177766	M10	DEC	–10	124
0131	00416	177765	M11	DEC	–11	1
0132	00417	002771	RNTER	DEF	2771B	25
0133	00420	177760	M16	DEC	–16	146
0134	00421	177715	M51	DEC	–51	3
0135	00422	177747	M25	DEC	–25	56
0136	00423	177740	M32	DEC	–32	124
0137	00424	177700	M64	DEC	–64	2
0138	00425	176400	M768	DEC	–768	
0139	00426	177753	M21	DEC	–21	6
0140	00427	177400	M256	DEC	–256	
0141	00430	002136	.INT2	DEF	2136B	2
0142	00431	010000	OB1	DEF	10000B	124
0143	00432	012000	OB2	DEF	12000B	2
0144	00433	014000	OB3	DEF	14000B	2
0145	00434	074302	K12	OCT	74302	2
0146	00435	100000	K13	OCT	100000	25
0147	00436	020040	K14	OCT	20040	2
0148	00437	000135	P93	DEC	93	25
0149	00440	001761	K15	DEF	1761B	2
0150	00441	001777	RSTK	DEF	1777B	
0151	00442	001400	K16	DEF	1400B	24
0152	00443	042101	K17	OCT	42101	2
0153	00444	040100	K18	OCT	40100	2
0154	00445	034000	K19	OCT	34000	2
0155	00446	004000	P2048	DEC	2048	2
0156	00447	001744	K20	DEF	1744B	12
0157	00450	001754	K21	DEF	1754B	12
0158	00451	000024	P20	DEC	20	4
0159	00452	000025	P21	DEC	21	46
0160	00453	000026	P22	DEC	22	4
0161	00454	000027	P23	DEC	23	4
0162	00455	000110	P72	DEC	72	35
0163	00456	001757	K22	DEF	1757B	4
0164	00457	000032	P26	DEC	26	45
0165	00460	016373	ENDSR	DEF	16373B	5
0166	00461	125252	CKBD2	OCT	125252	2
0167	00462	000153	P107	DEC	107	46
0168	00463	051505	K23	OCT	51505	2
0169	00464	000137	GAZ	OCT	137	

BASE PAGE 205B—554B 1400B—1777B

0170	00465	000141	EOL	OCT	141	5
0171	00466	000140	BOL	OCT	140	5
0172	00467	000122	R	OCT	122	5
0173	00470	000072	P58	DEC	58	5
0174	00471	177377	M257	DEC	-257	5
0175	00472	037600	K24	OCT	37600	5
0176	00473	002627	.SPAC	DEF	2627B	2
0177	00474	016367	CNTAB	DEF	16367B	5
0178	00475	000073	ENSTM	OCT	73	56
0179	00476	016114	.TAB1	DEF	16114A	5
0180	00477	001740	K25	DEF	1740B	56
0181	00500	001600	APMA	DEF	1600B	5
0182	00501	001723	K26	DEF	1723B	1
0183	00502	177767	M9	DEC	-9	246
0184	00503	000050	P40	DEC	40	46
0185	00504	000061	P49	DEC	49	4
0186	00505	000132	P90	DEC	90	4
0187	00506	000150	P104	DEC	104	4
0188	00507	000152	P106	DEC	106	456
0189	00510	000242	P162	DEC	162	4
0190	00511	000154	P108	DEC	108	46
0191	00512	003700	K27	OCT	3700	4
0192	00513	174007	K28	OCT	174007	34
0193	00514	100263	K29	OCT	100263	4
0194	00515	003777	K30	OCT	3777	2
0195	00516	177770	M8	DEC	-8	4
0196	00517	000442	K31	OCT	442	4
0197	00520	003335	K32	DEF	1757	4
0198	00521	106535	K34	OCT	106535	45
0199	00522	052525	CKBD1	OCT	52525	236
0200	00523	007122	STLN	DEF	7122B	5
0201	00524	004561	XRI	DEF	4561B	4
0202	00525	007516	CKSTP	DEF	7516B	5
0203	00526	000035	P29	DEC	29	5
0204	00527	000055	P45	DEC	45	26
0205	00530	000056	P46	DEC	46	26
0206	00531	002771	RESDP	DEF	2771B	2
0207	00532	001537	K35	DEF	1537B	5
0208	00533	003265	RET1	DEF	3265B	2
0209	00534	006670	ENTER	DEF	6670B	5
0210	00535	001432	K36	DEF	1432B	45
0211	00536	001726	K37	DEF	1726B	4
0212	00537	000041	P33	DEC	33	235
0213	00540	016400	BRWM	DEF	16400B	2
0214	00541	000051	P41	DEC	41	6
0215	00542	001727	K38	DEF	1727B	6
0216	00543	002731	L2F	DEF	2731B	15
0217	00544	000053	P43	DEC	43	26
0218	00545	000054	P44	DEC	44	56
0219	00546	000042	P34	DEC	34	456
0220	00547	007453	RECH	DEF	7453B	5
0221	00550	177634	M100	DEC	-100	24
0222	00551	016347	OPTN	DEF	16347B	5
0223	00552	004575	SXJ	DEF	4575B	4
0224	00553	000133	RUN	OCT	133	5
0225	00554	001670	.IBUF	DEF	1670B	2

```
0001                    ASMB,A,L
0002*
0003*  MTABL
0004*
0005   00005                    ORG 5B
0006*
0007   00005 150000 C0    OCT 150000      END-OF-LINE MARKER
0008   00006 150000 C1    OCT 150000      SPECIAL MARKER FOR ENTER TYPE STMTS
0009   00007 067400 C2    OCT 67400       STORE
0010   00010 073400 C3    OCT 73400       CLEAR
0011   00011 105400 C4    OCT 105400      LIST
0012   00012 120021 C5    OCT 120021      LOAD
0013   00013 140000 C6    OCT 140000      CLEAR FOR OPTION BLOCKS
0014   00014 150000 C7    OCT 150000      ILLEGAL
0015   00015 126425 C10   OCT 126425      K15
0016   00016 126426 C11   OCT 126426      K16
0017   00017 002252 C12   OCT 2252        *(IMP)
0018   00020 142021 C13   OCT 142021      CLEAR FLAG
0019   00021 176000 C14   OCT 176000      GOTO FLAG (ROUTE TO OPTION BLOCK)
0020   00022 132401 C15   OCT 132401  OPTION #3
0021   00023 126427 C16   OCT 126427      K17
0022   00024 126430 C17   OCT 126430      K18
0023   00025 075021 C20   OCT 75021       NORMAL
0024   00026 077021 C21   OCT 77021       TRACE
0025   00027 101021 C22   OCT 101021      FIXED
0026   00030 103021 C23   OCT 103021      FLOAT
0027   00031 066021 C24   OCT 66021       ENTER
0028   00032 116021 C25   OCT 116021      DISPLAY
0029   00033 112021 C26   OCT 112021      SPACE
0030   00034 114021 C27   OCT 114021      PRINT
0031   00035 122021 C30   OCT 122021      RECORD
0032   00036 054021 C31   OCT 54021       JMP
0033   00037 124021 C32   OCT 124021      END
0034   00040 070021 C33   OCT 70021       RETURN
0035   00041 110021 C34   OCT 110021      IF
0036   00042 060021 C35   OCT 60021       GOSUB
0037   00043 106315 C36   OCT 106315      FLAG
0038   00044 105021 C37   OCT 105021      SET FLAG
0039   00045 126431 C40   OCT 126431      K19 SPACE
0040   00046 073021 C41   OCT 73021       STOP
0041   00047 052000 C42   OCT 52000       "
0042   00050 000146 C43   OCT 146         #
0043   00051 126420 C44   OCT 126420      K10 S
0044   00052 126421 C45   OCT 126421      K11 %
0045   00053 126422 C46   OCT 126422      K12 &
0046   00054 126423 C47   OCT 126423      K13 '
0047   00055 044035 C50   OCT 44035       (
0048   00056 046002 C51   OCT 46002       )
0049   00057 012210 C52   OCT 12210       MULTIPLY *
0050   00060 016167 C53   OCT 16167       ADDITION +
0051   00061 050043 C54   OCT 50043       ,
0052   00062 020167 C55   OCT 20167       SUBTRACT -
0053   00063 036000 C56   OCT 36000       .
0054   00064 014210 C57   OCT 14210       DIVIDE /
0055   00065 032000 C60   OCT 32000       0
0056   00066 034001 C61   OCT 34001       1
```

```
0057  00067  034002  C62  OCT  34002    2
0058  00070  034003  C63  OCT  34003    3
0059  00071  034004  C64  OCT  34004    4
0060  00072  034005  C65  OCT  34005    5
0061  00073  034006  C66  OCT  34006    6
0062  00074  034007  C67  OCT  34007    7
0063  00075  034010  C70  OCT  34010    8
0064  00076  034011  C71  OCT  34011    9
0065  00077  030315  C72  OCT  30315    R-REG
0066  00100  056001  C73  OCT  56001    ;
0067  00101  004146  C74  OCT  4146     <=
0068  00102  006146  C75  OCT  6146     =
0069  00103  010146  C76  OCT  10146    >
0070  00104  126424  C77  OCT  126424   K14 ?
0071  00105  062021  C100 OCT  62021    GOTO @
0072  00106  042000  C101 OCT  42000    A
0073  00107  042004  C102 OCT  42004    B
0074  00110  042010  C103 OCT  42010    C
0075  00111  130440  C104 OCT  130440   K20 D
0076  00112  130441  C105 OCT  130441   K21 E
0077  00113  130442  C106 OCT  130442   K22 F
0078  00114  130443  C107 OCT  130443   K23 G
0079  00115  130444  C110 OCT  130444   K24 H
0080  00116  130445  C111 OCT  130445   K25 I
0081  00117  130446  C112 OCT  130446   K26 J
0082  00120  130447  C113 OCT  130447   K27 K
0083  00121  130450  C114 OCT  130450   K28 L
0084  00122  130451  C115 OCT  130451   K29 M
0085  00123  132460  C116 OCT  132460   K30 N
0086  00124  132461  C117 OCT  132461   K31 O
0087  00125  132462  C120 OCT  132462   K32 P
0088  00126  132463  C121 OCT  132463   K33 Q
0089  00127  132464  C122 OCT  132464   K34 R
0090  00130  132465  C123 OCT  132465   K35 S
0091  00131  132466  C124 OCT  132466   K36 T
0092  00132  132467  C125 OCT  132467   K37 U
0093  00133  132470  C126 OCT  132470   K38 V
0094  00134  132471  C127 OCT  132471   K39 W
0095  00135  042014  C130 OCT  42014    X
0096  00136  042020  C131 OCT  42020    Y
0097  00137  042024  C132 OCT  42024    Z
0098  00140  107400  C133 OCT  107400   RUN PROGRAM
0099  00141  024315  C134 OCT  24315    SQRT
0100  00142  065401  C135 OCT  65401    EXECUTE
0101  00143  040000  C136 OCT  40000    ENT EXP ↑
0102  00144  026263  C137 OCT  26263    GAZINTA ←
0103  00145  077400  C140 OCT  77400    RECALL BGLNE
0104  00146  075400  C141 OCT  75400    DELETE ENLNE
0105  00147  071400  C142 OCT  71400    INSERT
0106  00150  101400  C143 OCT  101400   FORWARD
0107  00151  103400  C144 OCT  103400   BACK
0108  00152  150000  C145 OCT  150000   RESET
0109  00153  126401  C146 OCT  126401  OPTION #1
0110  00154  130401  C147 OCT  130401  OPTION #2
0111  00155  022211  C150 OCT  22211      U-
0112  00156  150000  C151 OCT  150000     ILLEGAL
```

```
0113   00157  136000  C152  OCT  136000   NULL
0114   00160  150000  C153  OCT  150000   SKIP NEXT CODE FOR UNCOMPILER
0115   00161  134000  C154  OCT  134000   ALPHA
0116   00162  126432  C155  OCT  126432   THE FOLLOWING ASCII CODES ARE SPARE
                                            CODES
0117   00163  126433  C156  OCT  126433   FOR THE OPTION BLOCKS.
0118   00164  126434  C157  OCT  126434
0119   00165  126435  C160  OCT  126435
0120   00166  126436  C161  OCT  126436
0121   00167  126437  C162  OCT  126437
0122   00170  130452  C163  OCT  130452
0123   00171  130453  C164  OCT  130453
0124   00172  130454  C165  OCT  130454
0125   00173  130455  C166  OCT  130455
0126   00174  130456  C167  OCT  130456
0127   00175  130457  C170  OCT  130457
0128   00176  132472  C171  OCT  132472
0129   00177  132473  C172  OCT  132473
0130   00200  132474  C173  OCT  132474
0131   00201  132475  C174  OCT  132475
0132   00202  132476  C175  OCT  132476
0133   00203  132477  C176  OCT  132477
0134   00204  140000  C177  OCT  140000   USER PROGRAM BOUNDARY CODE
0135                      END
* NO ERRORS*
```

(76)

I/O SUPERVISOR

```
0001                    ASMB,A,L
0003*
0004  00000                 ORG  0B
0005*
0006* START-UP AND KEYBOARD LINKAGE PLUS CONSTANT
0007*
0008  00000  164003        JMP  3B,I     GO TO INITIALIZATION ROUTINE
0009  00001  043060        OCT  43060    SYSTEM CONSTANT
0010  00002  000000        BSS  1
0011  00003  002127        DEF  .INT     LINK WORD TO THE INITIALIZATION
                                           ROUTINE
0012  00004  000000        BSS  1
0013  00205                 ORG  205B
0014*
0015* KEYCODE TO SYMBOL TABLE TRANSFORM.
0016*
0017  00205  000033  .BF5HDEC 27                LOD
0018  00206  000000        BSS  2
0019  00210  056127        OCT  56127    *1    CFG
0020  00211  000000        BSS  2
0021  00213  011427        DEC  4887     NOR   TRC
0022  00214  005407        DEC  2823     FXD   FLT
0023  00215  035517        DEC  15183    ENT   DSP
0024  00216  007477        DEC  3903     SPC   PRT
0025  00217  031467        DEC  13111    REC   JMP
0026  00220  021507        DEC  9031     END   RET
0027  00221  025047        DEC  10791    IF    GSB
0028  00222  051533        OCT  51533    FLG   SFG
0029  00223  000057        DEC  47             STP
0030  00224  000000        DEC  0        "     #
0031  00225  000000        BSS  2
0032  00227  000000        DEC  0        (     )
0033  00230  000000        DEC  0        *     +
0034  00231  000000        DEC  0        ,     -
0035  00232  000000        DEC  0        .     /
0036  00233  000000        DEC  0        0     1
0037  00234  000000        DEC  0        2     3
0038  00235  000000        DEC  0        4     5
0039  00236  000000        DEC  0        6     7
0040  00237  000000        DEC  0        8     9
0041  00240  044000        DEC  18432    R     ;
0042  00241  000000        DEC  0        <=    =
0043  00242  000000        DEC  0        >
0044  00243  017400        DEC  7936     GTO   A
0045  00244  000000        DEC  0        B     C
0046  00245  000000        BSS  10
0047  00257  000000        DEC  0        X     Y
0048  00260  000000        DEC  0        Z
0049  00261  000000        DEC  0        SQR
0050  00262  040000        DEC  16384    E     ASSIGN
```

I/O SUPERVISOR

```
0052*
0053*  EQU WORDS
0054*
0055   01625   .SE      EQU  1625B   SYSTEM SECURITY WORD
0056   01626   .IL1     EQU  1626B   SYSTEM OPTION BLOCK TABLE
0057   00321   .END1    EQU  321B    SYSTEM END ROUTINE LINK WORD
0058   01556   AP1      EQU  1556B   STACK POINTER 1
0059   01557   AP2      EQU  1557B   STACK POINTER 2
0060   01546   CMA      EQU  1546B   PROGRAM MEMORY ADDRESS
0061   01542   ENDS     EQU  1542B   PRESENT LOCATION OF THE LAST WORD OF
                                      PROGRAM
0062   01377   MAW      EQU  1377B   UPPER LIMIT OF R/W MEMORY
0063   01550   .STPT    EQU  1550B   LAST RESULT TO BE PLACED IN THE DISPLAY
0064   01744   AR1      EQU  1744B   MATH REGISTER 1
0065   01754   AR2      EQU  1754B   MATH REGISTER 2
0066   01754   .REC.    EQU  AR2     LINK WORD FOR MAG CARD RECORD DRIVER
0067   01755   .SYM     EQU  AR2+1   LINK WORD FOR MAG CARD TURN OFF DRIVER
0068   01756   .LOD.    EQU  AR2+2   LINK WORD FOR MAG CARD LOAD DRIVER
0069   00315   INTRP    EQU  315B    SETUP ROUTINE FOR RUN TIME STACK
0070   00316   XEOL     EQU  316B    EXECUTE END OF LINE LINK WORD
0071   00320   RGTO     EQU  320B    RESET GTO'S AND GSB'S
0072   01572   NRSLT    EQU  1572B   HOLDS OLD ADDRESS OF .STPT
0073   01265   XRR      EQU  1265B   EXECUTE END-OF-LINE LINK WORD
0074   01352   RBAD     EQU  1352B
0075   01351   BAD      EQU  1351B   ROUTINE TO SET ERROR CODE AND FLAG
0076   00314   CMPLR    EQU  314B    COMPLIER ENTRY ADDRESS
0077   00525   U1       EQU  525B    UNCOMPLIER CONTINUE ADDRESS
0078   01544   SKEY     EQU  1544B   CONTAINS ACTUAL KEYCODE BEING PROCESSED
0079   01561   CODE     EQU  1561B   CONTAINS CODE WORD FOR KEYCODE BEING
                                      PROCESSED
0080   01575   QUOTE    EQU  1575B   FLAG USED TO INDICATE PRESENTS OF QUOTE
                                      FIELD
0081   01620   NCODE    EQU  1620B   SWITCH FOR OBTAINING NEXT MNEMONIC
                                      CHARACTER
0082   01702   GTS      EQU  1702B   GSB/GTO FLAG
0083*
0084*  SHARED R/W STORAGE
0085*
0086   01716   D        EQU  1716B   START OF SHARED AREA
0087   01716   I        EQU  D+0     I
0088   01717   J        EQU  D+1     J
0089   01720   K        EQU  D+2     K
0090   01721   L        EQU  D+3     L
0091   01722   M        EQU  D+4     M
0092   01723   N        EQU  D+5     N
0093   01724   P        EQU  D+6     P
0094   01725   R        EQU  D+7     R
0095   01726   S        EQU  D+8     S
0096   01727   T        EQU  D+9     T
0097   01730   I1       EQU  D+10    I1
0098   01731   I2       EQU  D+11    I2
0099   01732   I3       EQU  D+12    I3
0100   01733   I4       EQU  D+13    I4
0101   01734   I5       EQU  D+14    I5
0102   01735   I6       EQU  D+15    I6
0103   01736   I7       EQU  D+16    I7
0104   01737   I8       EQU  D+17    I8
0105   01730   TEMP     EQU  I1      USED BY THE MAG CARD FOR ADDRESS POINTER
0106   01731   WDCNT    EQU  I2      USED BY THE MAG CARD FOR WORD COUNT
0107*
```

I/O SUPERVISOR

```
0108*  NON-SHARED R/W STORAGE
0109*
0110   01660   .BUF    EQU  1660B  BEGINNING OF 16 WORD BUFFER AREA
0111   01700   .NS     EQU  1700B  START OF NON SHARED AREA
0112   01701   .WKC    EQU  .NS+1  TEMP STORAGE OF ENERED KEYCODE
0113   01703   .WPRT   EQU  .NS+3  WORD FOR FXD/FLT FORMAT CONVERSION
0114   01704   .WBUF   EQU  .NS+4  ADDRESS FOR STORAGE OF NEXT CHARACTER
0115   01705   .WMOD   EQU  .NS+5  SYSTEM STATUS WORD
0116   01706   .POS    EQU  .NS+6  HI/LO CHARACTER POSITION WORD
0117   01710   .END2   EQU  .NS+8  START OF END LINKAGE
0118   01560   OSW     EQU  1560B  SWITCH FOR ROUTING KEYCODES DURNING
                                     SYNTAX
0119   01543   LIM1    EQU  1543B  ADDRESS OF THE START OF THE PROGRAM
                                     AREA
0120*
0121   01563   TSW     EQU  1563B  STATUS WORD
0122   01604   NMA     EQU  1604B  CONTAINS ADDRESS OF NEXT LINE TO BE
                                     PROCESSED
0123   01615   BSTK    EQU  1615B  CONTAINS NUMBER OF WORDS FOR SYSTEM
                                     EXECUTION STACK
0124   01554   PCNT    EQU  1554B  CONTAINS NUMBER OF PARAMETERS DURING
                                     RECORD
0125   01555   PADD    EQU  1555B  CONTAINS NEXT PARAMETER ADDRESS
                                     DURING RECORD
0126   01250   XSTP    EQU  1250B  STOP ROUTINE. SETS STOP BIT IN .WMOD
```

I/O SUPERVISOR

```
0128*
0129*  ROM ADDRESS CONSTANTS
0130*
0131   00206                    ORG  206B
0132*
0133   00206  003000             DEF  .DSP1    LED DISPLAY DRIVER
0134   00207  001660  .DBUF      DEF  .BUF     START OF DISPLAY BUFFER
0135   00210  000000             BSS  1
0136   00211  101564  .TAB1      DEF  .IL1-42B,I  ADJUSTED ADDRESS OF OPTION BLOCK
                                                   ADDRESS TABLE
0137   00212  000203  .TAB3      DEF  .BF5H-2  KEYCODE TRANSLATION TABLE
0138*
0139   00225                    ORG  225B
0140   00225  003171             DEF  .LOD     MAG CARD LOAD ROUTINE
0141   00226  003043             DEF  .REC     MAG CARD RECORD ROUTINE
0142*
0143   00245                    ORG  245B
0144   00245  003470  .TAB5      DEF  L4-80    5x7 PATTERN TABLE
0145   00246  002100  .TAB6      DEF  .BF5F-1  STANDARD 3 LETTER KEYS
0146   00247  001626  .TAB7      DEF  .IL1     LINKAGE AREA FOR ADD-ON BLOCKS
0147   00250  002536             DEF  .PRT1    THERMO STRIP PRINTER DRIVER
0148   00251  002715             DEF  .DSP     DISPLAY MONITOR
0149   00252  002002             DEF  .BF31    STORE CHARACTER ROUTINE
0150   00253  002051             DEF  .BF5A    MNEMONIC GENERATOR
0151   00254  000000  SKCD       BSS  1
0152   00255  002227             DEF  L18A     CLEAR AND RESET BUFFER POINTERS
0153   00256  000430  .TAB4      DEF  K2-1     ADD-ON BLOCK ADDRESSES
0154*
0155   00276                    ORG  276B
0156   00276  020060  K15        OCT  20060    ASCII BLANK AND ASCII 0
0157   00277  002270             DEF  .FMT     FORMAT CONVERSION
0158   00300  002525             DEF  .BXFR    BUFFER TRANSFER
0159   00301  003271  .IL2       DEF  SYNTX    REJECT TABLE FOR OPTION BLOCKS
0160   00302  003273             DEF  EXEQX
0161   00303  000000             BSS  1
0162   00304  002241             DEF  .FLT     SET FLOAT MODE AND VALUE
0163   00305  002251             DEF  .FXD     SET FIXED MODE AND VALUE
0164   00306  002022             DEF  .BF5G    SET MNEMONIC LENGTH
0165   00307  000000             BSS  1
0166   00310  002260             DEF  .FMT2    DISPLAY RESULT—USE .SPST
0167*
0168   00324                    ORG  324B
0169   00324  002470             DEF  .RND-1   LINK ADDRESS FOR ROUNDING
0170   00473                    ORG  473B
0171   00473  002627             DEF  .SPAC    SPACE STRIP PRINTER
```

I/O SUPERVISOR

```
0173*
0174*
0175*  ROM CONSTANTS
0176
0177   00330                        ORG 330B
0178   00330  177762  DM14          DEC -14
0179   00331  000000                BSS 7
0180   00340  060141  BGNLE         OCT 060141  BEGINNING OF LINE CHARACTER
0181   00341  000400  DP256         OCT 400
0182   00342  000377  DP255         OCT 377
0183   00343  000220  DP144         OCT 220
0184   00344  000200  DP128         OCT 200
0185   00345  000177  DP127         OCT 177
0186   00346  000000  XJ            BSS 1
0187   00347  000120  DP80          OCT 120
0188   00350  000100  DP64          OCT 100
0189   00351  000000                BSS 4
0190   00355  000060  DP48          OCT 60
0191   00356  000000                BSS 2
0192   00360  000040  DP32          OCT 40
0193   00361  000037  DP31          OCT 37
0194   00362  000000                BSS 2
0195   00364  000020  DP16          OCT 20
0196   00365  000017  DP15          OCT 17
0197   00366  000016  DP14          OCT 16
0198   00367  000015  DP15          OCT 15
0199   00370  000014  DP12          OCT 14
0200   00371  000000                BSS 4
0201   00375  000007  DP7           OCT 7
0202   00376  000000                BSS 1
0203   00377  000005  DP5           OCT 5
0204   00400  000004  DP4           OCT 4
0205   00401  000003  DP3           OCT 3
0206   00402  000002  DP2           OCT 2
0207   00403  000001  DP1           OCT 1
0208   00404  000105  K6            OCT 105
0209   00405  177777  DM1           OCT 177777
0210   00406  177776  DM2           OCT 177776
0211   00407  177775  DM3           OCT 177775
0212   00410  177774  DM4           OCT 177774
0213   00411  177773  DM5           OCT 177773
0214   00412  000000                BSS 1
0215   00413  177771  DM7           OCT 177771
0216   00414  000000                BSS 1
0217   00415  177766  DM10          OCT 177766
0218   00416  177765  DM11          OCT 177765
0219   00417  002771                DEF RNTER
0220   00420  177760  DM16          OCT 177760
0221   00421  000000                BSS 2
0222   00423  177740  DM32          OCT 177740
0223   00424  177700  DM64          OCT 177700
0224   00425  176400  K16           OCT 176400  DELAY USED FOR STRIP PRINTER
0225   00426  000000                BSS 1
0226   00427  177400  DM256         OCT 177400
0227*
0228   00430  002136                DEF .INT2   POP MACHINE BUT DO NOT CLEAR MEMORY
```

I/O SUPERVISOR

```
0229  00431  010000  K2      OCT  10000,12000,14000  OPTION BLOCK ADDRESSES
      00432  012000
      00433  014000
0230  00434  074302  K5      SBR  7
0231  00435  100000  K7      OCT  100000
0232  00436  020040  K9      OCT  20040     2 ASCII BLANKS
0233  00437  000135  DP93    OCT  135
0234  00440  001761  K12     OCT  1761
0235  00441  001777  K13     OCT  1777
0236  00442  001400  K14     OCT  1400
0237  00443  042101  K1      ASC  1,DA
0238  00444  040100  K19     OCT  40100
0239  00445  034000  K20     OCT  34000
0240  00446  004000  K21     OCT  4000
0241  00447  001744  DAR1    DEF  AR1
0242  00450  001754  DAR2    DEF  AR2
0243  00451  000000          BSS  4
0244  00455  000110  DP72    DEC  72
0245  00456  000000          BSS  3
0246  00461  125252  K110    OCT  125252
0247  00462  000000          BSS  1
0248  00463  051505  K8      OCT  51505
0249  00464  000000          BSS  14
0250  00502  177767  DM9     DEC  -9
0251  00503  000000          BSS  10
0252  00515  037777  K22     OCT  37777
0253  00516  000000          BSS  4
0254  00522  052525  K11     OCT  52525
0255  00523  000000  STLN    BSS  1
0256  00524  000000          BSS  3
0257  00527  000055  DP45    OCT  55
0258  00530  000056  DP46    OCT  56
0259  00531  002731          DEF  L2F
0260  00532  000000          BSS  1
0261  00533  003265          DEF  L7A
0262  00534  000000          BSS  3
0263  00537  000041  DP33    OCT  41
0264  00540  016400  K3      OCT  16400
0265  00541  000000          BSS  2
0266  00543  002731          DEF  L2F
0267  00544  000053  DP43    OCT  53
0268  00545  000000          BSS  3
0269  00550  177634  DM100   OCT  177634
0270  00551  000000          BSS  3
0271  00554  001670  .IBUF   DEF  .BUF+8   FISRT WORD OF I/O BUFFER
```

I/O SUPERVISOR

```
0273*                                                    5X7 BIT PATTERNS
0274   03605                    ORG  3605B
0275   03605  000000  RLINK     BSS  3          MAG CARD DRIVER LINKAGE AREA
0276                            SUP
0277   03610  000000  L4       DEC  00000,00000,00125,00000,00000   BLANK  !
0278   03615  000026           DEC  00022,24596,00028,24596,00052   "      #
0279   03622  011142           DEC  04706,10852,32520,10771,09251   S      %
0280   03627  033000           DEC  13824,18792,13680,00512,01280   &      '
0281   03634  000000           DEC  00000,00065,15934,16640,00000   (      )
0282   03641  004010           DEC  02056,10760,07230,10760,02056   *      +
0283   03646  000010           DEC  00008,03336,03592,00008,00008   ,      -
0284   03653  000002           DEC  00002,00772,00776,00016,00032   .      /
0285   03660  037000           DEC  15872,17697,18815,20737,15872   0      1
0286   03665  021442           DEC  08994,17729,18761,18761,12598   2      3
0287   03672  006162           DEC  03186,05201,09297,32593,01102   4      5
0288   03677  017100           DEC  07744,10567,18760,18768,01632   6      7
0289   03704  033060           DEC  13872,18761,18761,18762,13884   8      9
0290   03711  000000           DEC  00000,13933,13934,00000,00000   :      ;
0291   03716  004424           DEC  02324,05396,08980,16660,00020   <=     =
0292   03723  040460           DEC  16688,08768,05189,02120,00048   >      ?
0293   03730  023077           DEC  09791,18760,20040,16712,15935   @      A
0294   03735  077476           DEC  32574,18753,18753,18753,13858   B      C
0295   03742  040577           DEC  16767,32585,16713,16713,15937   D      E
0296   03747  077476           DEC  32574,18497,18497,18501,16455   F      G
0297   03754  077400           DEC  32512,02113,02175,02113,32512   H      I
0298   03761  001177           DEC  00639,00264,00276,00290,32321   J      K
0299   03766  077577           DEC  32639,00288,00280,00288,00383   L      M
0300   03773  077476           DEC  32574,04161,02113,01089,32574   N      O
0301   04000  077476           DEC  32574,18497,18501,18498,12349   P      Q
0302   04005  077462           DEC  32562,18505,19529,19017,12582   R      S
0303   04012  040176           DEC  16510,16385,32513,16385,16510   T      U
0304   04017  070176           DEC  28798,03073,00782,03073,28798   V      W
0305   04024  061540           DEC  25440,05136,02063,05136,25440   X      Y
0306   04031  041410           DEC  17160,17695,18704,20767,24864   Z      PI
0307   04036  001020           DEC  00528,16152,08220,08216,08208   SQR    INV
0308   04043  010010           DEC  04104,08200,32298,08220,04104   ↑      ASSIGN
0309   04050  000177           DEC  00127,00008,00008,00008,00008   BOL    EOL
```

I/O SUPERVISOR
BUFFER STORING MNEMONIC TRANSFORM
ROUTINES

```
0311   02001                    ORG  2001B
0312*
0313*****************************************
0314*
0315*  .BF3* IS USED TO STORE THE 8 BIT CHARACTER FOUND IN
0316*  LOCATION K INTO THE NEXT AVAILABLE LOCATION OF THE I/O,
0317*  AND DISPLAY BUFFERS RESPECTIVELY.
0318*
0319   02001  031720        STA  K          SAVE CHARACTER
0320   02002  025706 .BF31   LDB  .POS       GET POSITION CODE
0321   02003  074710        SZB  L6H        IF ZERO EXIT ROUTINE
0322   02004  074744        SBL  1          ELSE SHIFT CODE LEFT ONE
0323   02005  035706        STB  .POS         AND SAVE
0324   02006  121704        LDA  .WBUF,I    GET BUFFER WORD
0325   02007  074113        SBP  *+2        IF CHARACTER TO BE STORED IN HI PART
0326   02010  070346        RAR  8            OF WORD POSITION BUFFER WORD TO
                                               LOW PART
0327   02011  050427        AND  DM256      MASK OUT LOW PART OF WORD
0328   02012  041720        IOR  K          INCLUDE NEW CHARACTER TO BE STORED
0329   02013  074113        SBP  *+2        REPOSITION IF CHARACTER
0330   02014  070346        RAR  8            IS BEING STORED IN THE HIGH PART
                                               OF WORD
0331   02015  131704        STA  .WBUF,I    STORE WORD BACK INTO THE BUFFER
0332   02016  074150        SZB  *+3
0333   02017  074112        SBM  *+2        IF CHARACTER WAS IN THE LOW PART
0334   02020  045704        ISZ  .WBUF        INCREMENT BUFFER ADDRESS
0335   02021  170402 L6H    RET             RETURN
0336*
0337*  .BF5G DETERMINES THE NUMBER OF LETTERS -1 NEEDED FOR THE
0338*  TRANSLATION OF A KEYCODE TO A DISPLAYED MNEMONIC. CALLING
0339*  SEQUENCE:
0340*
0341*      LDA KEYCODE
0342*      JSM .BF5G
0343*
0344*  RETURN IS MADE WITH N-1 IN A. IF TRANSLATION IS NECESSARY
0345*  B WILL CONTAIN MNEMONIC WORD, ELSE B=0.
0346*
0347*  CHECK KEYCODE CLASS FOR OPTION BLOCK.
0348*
0349   02022  160254 .BF5G  JSM  SKCD,I    GET KEYCODE CLASS
0350   02023  010403        CPA  DP1        OPTION BLOCK?
0351   02024  066045        JMP  L6A          YES, GO TO L6A
0352*
0353*  KEYCODE IS FOR BASIC MACHINE. GET COUNT AND MNEMONIC.
0354*
0355   02025  021544        LDA  SKEY       GET KEYCODE
0356   02026  070002        SAR  1            DIVIDE BY 2
0357   02027  000212        ADA  .TAB3          AND ADD .TAB3 ADDRESS
0358   02030  070517        LDA  A,I        LOAD SYMBOL INFORMATION
0359   02031  025544        LDB  SKEY       GET KEYCODE
0360   02032  074056        CMB               AND SKIP
0361   02033  074111        SLB  *+2            IF EVEN
0362   02034  070342        SAR  8          REPOSITION SYMBOL WORD
0363   02035  050342        AND  DP255      MASK OUT HIGH PART
0364   02036  070137        LDB  A          SET SYMBOL LENGTH-1
0365   02037  050401        AND  DP3          IN A
0366   02040  074210        SZB  *+4        IF B=0, NO TRANSLATION NECESSARY —
                                               RETURN
```

```
0367  02041  074042          SBR  2        GET RELATIVE ADDRESS
0368  02042  004246          ADB  .TAB6    ADD TABLE ADDRESS
0369  02043  074537          LDB  B,I      SET B=MNEMONIC CODE
0370  02044  170402          RET           RETURN
0371*
0372* TRANSFER CONTROL TO OPTION BLOCK. THE A REGISTER CONTAINS
0373* THE RELATIVE KEYCODE.
0374*
0375  02045  021561 L6A      LDA  CODE     SET RELATIVE KEYCODE FOR
0376  02046  050365          AND  DP15       OPTION BLOCK
0377  02047  004211          ADB  .TAB1    TRANSFER TO MNEMONIC GENERATOR OF
                                            OPTION BLOCK
0378  02050  074737          JMP  B,I      IMPLIED RETURN
0379*
0380* .BF5A TAKES THE KEY CODE STORED IN K AND GENERATES A 1
0381* TO 4 CHARACTER SEQUENCE. CALLING SEQUENCE:
0382*
0383*      ST*  K             KEYCODE TO BE TRANSFORMED
0384*      JSM  .BF5A
0385*
0386* RETURN WILL BE MADE WITH THE NEXT SYMBOL TO BE STORED IN K. M IS
0387* SET TO N-1 WHERE N IS THE TOTAL NUMBER OF SYMBOLS TO BE STORED.
0388* IF M≠0, CALLING .BF5B WILL GIVE THE NEXT SYMBOL UNTIL M GOES
0389* TO ZERO.
0390*
0391  02051  021720 .BF5A    LDA  K        LOAD A WITH KEYCODE
0392  02052  062022          JSM  .BF5G    GET LENGTH AND MNEMONIC CODE
0393  02053  071452          SAM  L6A-1    IF A NEGATIVE, RETURN
0394  02054  070076          TCA           SET COUNT NEGITIVE
0395  02055  031722          STA  M          AND SAVE IN M
0396  02056  075310          SZB  L6A-1    IF B=0, THEN NO TRANSLATION NECESSARY
0397*
0398* TRANSLATION IS NECESSARY. POSITION MNEMONIC WORD AND
0399* GET FIRST CHARACTER.
0400*
0401  02057  035717          STB  J        SAVE MNEMONIC CODE
0402  02060  074117          LDA  B        SET A←B
0403  02061  070446          RAR  10       POSITION 1ST 5 BIT CODE IN LOW PART OF A
0404  02062  066065          JMP  .BF5D    CONTINUE
0405*
0406* .BF5B IS THE CONTINUATOR FOR A MULTIPLE CHARACTER SEQUENCE.
0407*
0408  02063  021721          LDA  L        GET MNEMONIC CODE
0409  02064  070506          RAR  11       POSITION NEXT 5 BIT CODE IN LOW PART OF A
0410  02065  031721 .BF5D    STA  L        SAVE MNEMONIC CODE
0411  02066  050361          AND  DP31     SAVE LOW 5 BITS
0412  02067  000350          ADA  DP64     FORM ASCII CODE
0413  02070  031720          STA  K          AND STORE IN K
0414  02071  170402          RET           RETURN
0415*
0416  02072  045722 .BF5B    ISZ  M        INCREMENT AND CHECK MNEMONIC COUNT
0417  02073  066063          JMP  .BF5D-2  NOT DONE. FORM NEXT CHARACTER
0418*
0419* IF NO SPACE SUPPRESION, ADD A SPACE, ELSE GET LAST
0420* CHARACTER
0421*
0422  02074  021717          LDA  J        SET A←J
```

I/O SUPERVISOR

```
0423  02075  071312          SAM .BF5D-2  IF A<0 THEN SUPPRESS SPACE AND FORM LAST
                                          CHAR.
0424  02076  020360          LDA DP32     ELSE.
0425  02077  066070          JMP .BF5D+3  FORM SPACE
0426*
0427* THE FOLLOWING IS THE TABLE CONTAINING THE THREE
0428* LETTER NMEMONICS FOR THE STANDARD KEY CODES
0429*
0430  02100  002072  K10   DEF .BF5B
0431  02101  014624  .BF5F DEC 6548      FLT
0432  02102  015404        DEC 6916      FXD
0433  02103  047003        DEC 19971     SPC
0434  02104  034762        DEC 14834     NOR
0435  02105  051103        DEC 21059     TRC
0436  02106  030744        DEC 12772     LOD
0437  02107  017217        DEC 7823      GTO
0438  02110  012704        DEC 5572      END
0439  02111  017142        DEC 7778      GSB
0440  02112  022300        OCT 22300     IF
0441  02113  047220        DEC 20112     SJP
0442  02114  044243        DEC 18595     REC
0443  02115  024660        OCT 24660     JMP
0444  02116  012724        DEC 5588      ENT
0445  02117  041124        DEC 16980     PRT
0446  02120  112000        DEC -27648    E
0447  02121  044264        DEC 18612     RET
0448  02122  144000        DEC -14336    R
0449  02123  011160        DEC 4720      DSP
0450  02124  014607        OCT 14607     FLG
0451  02125  006307        DEC 3271      CFG
0452  02126  046307        DEC 19655     SFG
```

I/O SUPERVISOR

```
0454*
0455*  INITIATION ROUTINE
0456*
0457*  DISABLE KEYBOARD AND CLEAR ALL OF SYSTEM R/W MEMORY.
0458*
0459   02127  172741  .INT    STF  1          DISABLE KEYBOARD
0460   02130  024442          LDB  K14        GET ADDRESS OF 1ST WORD OF SYSTEM R/W
                                                MEMORY
0461   02131  070742          SAR  16         SET A←0
0462   02132  015377          CPB  MAW        DONE?
0463   02133  066136          JMP  *+3          YES, JUMP OUT OF LOOP
0464   02134  074557          STA  B,I        STORE <B>←0
0465   02135  077670          RIB  *-3        RECYCLE
0466*
0467*  RESET JSM STACK POINTER AND SET FLOAT 9 AND FIXED 0.
0468*
0469   02136  020440  .INT2   LDA  K12      SET A←1761B (FIRST WORD USED BY RET STACK)
0470   02137  130441          STA  K13,I      SAVE IN STACK POINTER (1777)
0471   02140  020343          LDA  DP144      SET .WPRT TO BE FLOAT 9
0472   02141  031703          STA  .WPRT        WHEN THE MACHINE IS TURNED ON
0473*
0474*  DO THE FOLLOWING
0475*
0476*  1. SET LIM1
0477*  2. SET .STPT TO ADDRESS OF A.
0478*  4. SET OSW
0479*  5. SET UNCTR
0480*  6. SET NCODE
0481*
0482   02142  020540          LDA  K3         SET LIM1 TO POINT TO
0483   02143  031543          STA  LIM1         16400B—START OF USER R/W MEMORY
0484   02144  020405          LDA  DM1        SET FOR NON NUMERIC ANSWER
0485   02145  031550          STA  .STPT
0486   02146  031572          STA  NRSLT
0487   02147  020314          LDA  CMPLR      GET NOMAL COMPILER ENTRY ADDRESS
0488   02150  031560          STA  OSW        SET IT INTO SWITCH OSW
0489   02151  002210Ø         LDA  K1Ø        GET NORMAL MNEMONIC CONTINUE ADDRESS
0490   02152  031620          STA  NCODE      SET IT INTO SWITCH NCODE
0491   02153  020455          LDA  DP72       BSSTK←72
0492   02154  031615          STA  BSTK       SET SYSTEM EXECUTION STACK SIZE
0493*
0494*  ADD DAISY CHAIN END LINK WORD.
0495*
0496   02155  020321          LDA  .END1      GET NORMAL END ENTRY ADDRESS
0497   02156  031710          STA  .END2      SET IT INTO LINK WORD .END2
0498*
0499*  SET UP LINKAGE FOR ADD-ON BLOCKS. IF BLOCK IS PRESENT
0500*  THEN DO THE FOLLOWING
0501*
0502*  1. SET CONFIGURATION CODE.
0503*  2. SET MNEMONIC LINK
0504*  3. SET SYNTAX LINK.
0505*  4. SET EXECUTION LINK.
0506*  5. HAVE ADD-ON BLOCK DO ITS INITIATION AND RETURN.
0507*
0508*  ELSE
0509*
```

I/O SUPERVISOR

```
0510*   1. SET REJECT SYNTAX LINK.
0511*   3. REJECT MNEMONIC LINK.
0512*
0513*   THE ADD-ON BLOCK MAY NOT USE THE FOLLOWING SHARED R/W MEMORY
0514*   WORDS DURNING ITS INITIATIONS: L,M,& N.
0515*
0516*   FIRST SET ALL THE REJECT ADDRESSES.
0517*
0518  02157  020247        LDA  .TAB7   FORM ADDRESS OF OPTION BLOCK
0519  02160  000401        ADA  DP3       LINKAGE AREA
0520  02161  031722        STA  M           AND SAVE IN M
0521  02162  020301        LDA  .IL2    A←SYNTAX REJECT LINK WORD
0522  02163  062222        JSM  .IA     STORE A IN THE TABLE
0523  02164  020302        LDA  .IL2+1  A←EXECUTION REJECT LINK WORD
0524  02165  062222        JSM  .IA     STORE A IN THE TABLE
0525  02166  020303        LDA  .IL2+2  A←MNEMONIC REJECT LINK WORD
0526  02167  062222        JSM  .IA     STORE A IN THE TABLE
0527*
0528*   NOW SET ADD-ON BLOCK LINKAGE WHERE NEEDED.
0529*
0530  02170  020247        LDA  .TAB7   STORE ADDRESS OF OPTION BLOCK
0531  02171  031722        STA  M         TABLE IN M
0532  02172  020407        LDA  DM3     SET
0533  02173  031723        STA  N         N←—3
0534  02174  020256        LDA  .TAB4   SET
0535  02175  000403        ADA  DP1       L←.TAB4+1
0536  02176  031721        STA  L           (LINK ADDRESS TO 1ST OPTION BLOCK)
0537  02177  121721  .IB   LDA  L,I    GET BASE ADDRESS OF OPTION BLOCK
0538  02200  031720        STA  K         AND SAVE
0539  02201  025722        LDB  M       SET B←OPTION BLOCK TABLE ADDRESS
0540  02202  020410        LDA  DM4     SET J←—4
0541  02203  031717        STA  J
0542  02204  121720  L6F   LDA  K,I     READ OPTION BLOCK WORD
0543  02205  070410        SZA  L6B     IF ZERO, SKIP TO NEXT BLOCK
0544  02206  141721        IOR  L,I     ELSE ADD BASE ADDRESS
0545  02207  074557        STA  B,I     AND SAVE IN THE TABLE
0546  02210  045720        ISZ  K       INCREMENT OPTION BLOCK ADDRESS
                                          POINTER
0547  02211  000401        ADB  DP3     ADVANCE TABLE ADDRESS BY 3
0548  02212  045717        ISZ  J       LINKAGE FINISHED?
0549  02213  060204        JMP  L6F       NO, CONTINUE TO LOOP
0550  02214  161720        JSM  K,I       YES, LET OPTION BLOCK DO IT'S
                                            INITIALIZATION
0551  02215  045722  L6B   ISZ  M       INCREMENT TABLE ADDRESS
0552  02216  045721        ISZ  L       INCREMENT OPTION BLOCK LINK WORD
0553  02217  045723        ISZ  N       ALL THREE BLOCKS DONE?
0554  02220  060177        JMP  .IB       NO, CONTINUE TO LOOP
0555  02221  164317        JMP  317B,I    YES, GO TO SYSTEM END ROUTINE
0556*
0557  02222  024406  .IA   LDB  DM2     SET B←—2
0558  02223  131722        STA  M,I     <M>←(A)
0559  02224  045722        ISZ  M       INCREMENT M
0560  02225  077730        RIB  *—2     IF B≠0, THEN B←B+1 AND LOOP
0561  02226  170402        RET          ELSE RETURN
0562*
0563*   L18A IS USED TO RESET EITHER THE DISPLAY OR I/O
0564*   BUFFER WITH BLANKS. CALLING SEQUENCE
0565*
```

I/O SUPERVISOR

FORMAT ROUTINES

```
0566*       LDA (BUFFER ADDRESS)
0567*       JSM L18A
0568*
0569  02227 031722 L18A STA  M     SAVE BUFFER ADDRESS
0570  02230 031704       STA  .WBUF   IN M AND .WBUF
0571  02231 020436       LDA  K9    A←-2 ASCII BLANKS
0572  02232 024413       LDB  DM7   B←-7
0573  02233 131722       STA  M,I   STORE BLANKS
0574  02234 045722       ISZ  M       IN ALL 8
0575  02235 077730       RIB  *-2       BUFFER WORDS
0576  02236 020522       LDA  K11   RESET
0577  02237 031706       STA  .POS    HI/LOW POINTER (52525)
0578  02240 170402       RET        RETURN
```

I/O SUPERVISOR

```
0580*  .FLT AND .FXD SET THE DISPLAY FORMAT FOR THE 9000A.
0581*  CALLING SEQUENCE IS:
0582*
0583*       LDA PARAMETER COUNT
0584*       LDB SIGNLE DIGIT.
0585*       JSM .FLT (OR .FXD)
0586*
0587*  IF B IS NEGITIVE THEN DON'T CHANGE SETTING OF N. OTHERWISE SET
0588*  N ACCORDING THE VALUE FOUND IN THE B REGISTER,
0589*
0590   02241  021703  .FLT   LDA  .WPRT   CLEAR FIXED FLAG
0591   02242  070073         SAP  *+1,C
0592   02243  074552         SBM  L6C     IF NO DIGIT SENT, GO TO L6C
0593   02244  070146         RAR  4       ELSE POSITION
0594   02245  050420         AND  DM16      AND SET NEW VALUE
0595   02246  074217         IOR  B           N FOR FLOAT PART OF .WPRT
0596   02247  070546         RAR  12
0597   02250  066256         JMP  L6C     GO TO L6C
0598   02251  021703  .FXD   LDA  .WPRT   SET FIXED
0599   02252  072052         SAM  *+1,S       FLAG
0600   02253  074152         SBM  L6C     IF NO DIGIT SENT, GO TO L6C
0601   02254  050420         AND  DM16    ELSE SET NEW VALUE OF
0602   02255  074217         IOR  B         N FOR FIXED PART OF .WPRT
0603   02256  031703  L6C    STA  .WPRT   SAVE .WPRT
0604   02257  170402         RET          RETURN
```

I/O SUPERVISOR

```
0606*
0607*  .FMT2 IS USED BY THE SYSTEM FOR REBUILDING THE PREVIOUS
0608*  RESULT.  WHEN CALLED, .FMT2 WILL TAKE THE LAST RESULT OF
0609*  THE STACK AND IF IT IS A NUMERIC RESULT, IT WILL REBUILD
0610*  THE RESULT ACCORDING TO THE PRESENT FORMAT. OTHERWISE,
0611*  IT WILL TRANSFER THE ASCII STRING FOUND IN .IBUF FOR DISPLAY.
0612*
0613  02260 021550 .FMT2 LDA .STPT   A←ADDRESS OF LAST RESULT
0614  02261 070744       SAL 1       SHIFT A LEFT 1
0615  02262 070132       SAM *+2,C   IF A<0 THEN RESULT IS ALPHA—SKIP FMT
                                       ROUTINE
0616  02263 062267       JSM .FMT-1  ELSE CONVERT LAST NUMERIC RESULT
0617  02264 020554       LDA .IBUF   TRANSFER I/O BUFFER
0618  02265 024207       LDB .DBUF     TO DISPLAY BUFFER
0619  02266 066525       JMP .BXFR   IMPLIED RETURN
```

```
0621*
0622**********************************************
0623*
0624*  THE FORMATTER CONVERTS THE DATA WORD SPECIFIED BY THE ADDRESS
0625*  FOUND IN THE A REGISTER ACCORDING TO THE FORMAT STORED IN THE
0626*  9820A. THE OUTPUT IS PLACED IN THE I/O BUFFER ON RETURN.
0627*
0628************************************************************
0629*
0630*  RESET I/O REGISTER.
0631*
0632  02267 070002       SAR 1       REPOSTION ADDRESS FOR .FMT2 ROUTINE
0633  02270 031734 .FMT  STA 15      SAVE ADDRESS OF DATA WORD
0634  02271 020554       LDA .IBUF   RESET I/O BUFFER
0635  02272 062227       JSM L18A      AND IT'S POINTERS
0636*
0637*  CHECK FOR FIXED OF FLOAT FORMAT
0638*
0639  02273 021703       LDA .WRPT   A←.WPRT
0640  02274 070113       SAP FLOAT   IF A(15)≠1 THEN FLOAT CONVERSION ROUTINE
0641  02275 066353       JMP FIXED   ELSE FIXED CONVERSION ROUTINE
0642*
0643*  FLOATING POINT ROUTINE. USE FOLLOWING FORMAT.
0644*
0645*      SX.XXXXXXXXXESXX
0646*
0647*  IF FLOAT N IS LESS THEN 9, THEN S IS MOVED TO THE RIGHT
0648*  (9-N) PLACES AND THE HOLE IS FILLED WITH BLANKS.
0649*
0650*  FIRST ROUND THE (N+2) DIGIT OF THE MANTISSA.
0651*
0652  02276 021703 FLOAT LDA .WPRT   A←.WPRT
```

I/O SUPERVISOR

```
0653  02277  070146          RAR 4          GET VALUE OF FLOAT N
0654  02300  050365          AND DP15         AND SAVE
0655  02301  031732          STA I3             IN I3
0656  02302  000502          ADA DM9        STORE 9-(FLOAT N)
0657  02303  062440          JSM L24A         BLANKS IN I/O BUFFER
0658  02304  021734          LDA I5         ROUND DATA WORD 15
0659  02305  025732          LDB I3           IN THE N+2 DIGIT
0660  02306  004402          ADB DP2
0661  02307  062471          JSM .RND
0662*
0663* CHECK THE VALUE OF THE EXPONENT AND IF IT IS OUTSIDE OF THE
0664* RANGE OF +99, THEN USE THE ORIGINAL NUMBER WITHOUT ROUNDING.
0665*
0666  02310  021754          LDA AR2        GET AND POSITION EXPONENT
0667  02311  070340          AAR 8
0668  02312  000550          ADA DM100      IF EXPONENT >+99
0669  02313  070212          SAM L24H
0670  02314  021734          LDA I5         THEN TRANSFER ORIGINAL
0671  02315  024450          LDB DAR2         NUMBER INTO AR2
0672  02316  170004          XFR
0673  02317  070075  L24H    SEC *+1,C      SET E←0 TO
0674  02320  062447          JSM L24B         STORE SIGN OF MANTISSA IN TO THE BUFFER
0675  02321  020403          LDA DP1        STORE 1ST DIGIT
0676  02322  062457          JSM L24C         INTO THE BUFFER
0677  02323  020530          LDA DP46       STORE DECIMAL POINT
0678  02324  062001          JSM .BF3I-1      INTO THE BUFFER
0679  02325  021732          LDA I3         IF I3≠0 THEN
0680  02326  070110          SZA *+2          STORE N DIGITS
0681  02327  062457          JSM L24C           INTO THE BUFFER
0682  02330  020404          LDA K6         STORE ASCII E
0683  02331  062001          JSM .BF3I-1      INTO THE BUFFER
0684  02332  072054          SES *+1,S      SET E←1 TO
0685  02333  062447          JSM L24B         STORE SIGN OF THE EXPONENT INTO THE
                                              BUFFER
0686*
0687* CONVERT AND STORE MANTISSA.
0688*
0689  02334  021754          LDA AR2        GET EXPONENT AND
0690  02335  070340          AAR 8            POSITION
0691  02336  070113          SAP *+2        IF EXPONENT<0
0692  02337  070076          TCA              THEN A←-A
0693  02340  024355  L24G    LDB DP48       STORE ASCII "0"
0694  02341  035720          STB K            IN K
0695  02342  031722          STA M          SAVE LAST RESULT
0696  02343  000415          ADA DM10       DECREMENT A BY 10
0697  02344  070152          SAM *+3        EXIT LOOP IF A<0
0698  02345  045720          ISZ K          ELSE INCREMENT K
0699  02346  066342          JMP *-4          AND CONTINUE TO LOOP
0700  02347  062002          JSM .BF3I      STORE 1ST DIGIT OF EXPONENT
0701  02350  021722          LDA M          FORM 2ND ASCII DIGIT OF
0702  02351  000355          ADA DP48       THE EXPONENT AND STORE IT IN THE BUFFER
0703  02352  066001          JMP .BF3I-1    IMPLIED RETURN
```

I/O SUPERVISOR

```
0705*
0706* IN FIXED MODE, THE FORMAT CONTAINS A SIGN (BLANK FOR PLUS),
0707* A DECIMAL POINT, AND UP TO TEN DIGITS. LEADING ZEROES UP
0708* TO THE DECIMAL POINT ARE SUPPRESSED. THE ENTIRE FORMAT IS
0709* RIGNT JUSTIFIED IN THE RIGHT MOST 12 COLUMNS. THE DECIMAL
0710* POINT IS IN COLUMN N+1, WHERE N IS THE DECIMAL SETTING
0711* OVER THE RANGE 0 TO 9.
0712*
0713* FIRST CHECK TO SEE THAT DATA WORD WILL FIT INTO FIXED FORMAT.
0714* IF NOT, USE FLOAT FORMAT.
0715*
0716  02353 050365 FIXED AND DP15    GET FIXED IN
0717  02354 031731       STA  I2       AND SAVE IN I2
0718  02355 121734       LDA  15,I     GET AND POSITION
0719  02356 070340       AAR  8          EXPONENT
0720  02357 000403       ADA  DP1      ADD 1 TO DETERMINE NUMBER OF DIGITS TO
0721  02360 031733       STA  I4         THE LEFT OF THE DECIMAL POINT AND SAVE
                                           IN I4
0722  02361 001731       ADA  I2       ADD NUMBER OF DIGITS TO RIGHT OF THE
0723  02362 031732       STA  I3         DECIMAL POINT TO BE DISPLAYED AND SAVE
                                           TOTAL DIGITS FOR DISPLAY
0724  02363 000416       ADA  DM11     IF THE ABOVE NUMBER IS > 10
0725  02364 070112       SAM  *+2
0726  02365 066276       JMP  FLOAT    THEN THE FLOAT CONVERSION MUST BE USED
0727*
0728* ROUND THE (N+1) DIGIT AFTER THE DECIMAL POINT. IF MANTISSA
0729* OVERFLOW OCCURES, INCREMENT I3 & 14.
0730*
0731  02366 021734       LDA  I5       ROUND DATA POINT 15
0732  02367 025732       LDB  I3         IN DIGIT I3+1
0733  02370 004403       ADB  DP1          (1ST DIGIT NOT TO BE DISPLAYED)
0734  02371 062471       JSM  .RND
0735  02372 074253       SBP  L6D      IF NO CARRY OCCURED ON ROUNDING, GO TO
                                           L6D
0736  02373 045732       ISZ  I3       ELSE INCREMENT NUMBER OF DIGITS TO
0737  02374 070055       SEC  *+1        THE LEFT OF THE DECIMAL POINT
0738  02375 045733       ISZ  I4       INCREMENT TOTAL NUMBER OF DIGITS TO
0739  02376 070055       SEC  *+1        BE DISPLAYED
0740*
0741* SET UP LEFT MOST BLANK FIELD AND STORE SIGN OF MANTISSA.
0742*
0743  02377 025733 L6D   LDB  I4       B← NUMBER OF DIGITS TO THE LEFT OF THE
                                           DECIMAL POINT
0744  02400 021732       LDA  I3       A←TOTAL NUMBER OF DIGITS TO BE DISPLAYED
0745  02401 074113       SBP  *+2      IF B< 0 THEN SET A ← I2
0746  02402 021731       LDA  I2         (NUMBER OF DIGITS TO RIGHT OF DECIMAL
                                           POINT)
0747  02403 025731       LDB  I2       IF I2=0 (INTEGER DISPLAY)
0748  02404 076110       RZB  *+2        THEN DECREMENT VALUE
0749  02405 000405       ADA  DM1          OF THE NUMBER OF CHARACTERS TO BE
                                             DISPLAYED
0750  02406 000330       ADA  DM14     DETERMINE THE NUMBER OF LEADING BLANKS
0751  02407 062440       JSM  L24A       AND THEN STORE THEM IN THE BUFFER
0752  02410 070075       SEC  *+1,C    SET E←0 TO
0753  02411 062447       JSM  L24B       STORE SIGN OF THE MANTISSA IN THE BUFFER
0754*
0755* STORE DIGITS TO RIGHT OF DECIMAL POINT IF ANY. THEN
0756* ADD DECIMAL POINT.
0757*
0758  02412 021733       LDA  I4       IF I4>0
0759  02413 070210       SZA  *+4
0760  02414 070152       SAM  *+3
```

```
0761  02415 062457       JSM  L24C     THEN STORE 14 DIGITS IN THE BUFFER
0762  02416 066423       JMP  L6G        AND SKIP THE INTEGER TEST
0763  02417 021731       LDA  I2       IF I2 ≠ 0 (NON-INTEGER MODE)
0764  02420 072150       RZA  L6G        THEN GO TO L6G
0765  02421 020276       LDA  K15      ELSE STORE AN INTEGER 0
0766  02422 131704       STA  .WBUF,I          IN THE BUFFER
0767  02423 020530 L6G   LDA  DP46     STORE A DECIMAL POINT
0768  02424 062001       JSM  .BF3I-1    IN THE BUFFER
0769*
0770* STORE ZEROES AS NEEDED AND THEN DIGITS TO RIGHT OF
0771* DECIMAL POINT.
0772*
0773  02425 021733       LDA  I4       IF I4 NOT < 0
0774  02426 070413       SAP  L6E        THEN GO TO L6E
0775  02427 031721       STA  L        SAVE THE VALUE OF THE NUMBER OF
                                           ZEROES TO BE STORED
0776  02430 021706       LDA  .POS     BUFFER FULL?
0777  02431 070650       SZA  L24E       YES, EXIT ROUTINE
0778  02432 020355       LDA  DP48       NO, STORE ANOTHER
0779  02433 062001       JSM  .BF3I-1      ASCII ZERO IN THE BUFFER
0780  02434 045721       IS   L        MORE ZEROS NEEDED?
0781  02435 066430       JMP  *-5        YES, CONTINUE TO LOOP
0782  02436 021731 L6E    LDA  I2         NO, STORE 12 DIGITS
0783  02437 066457       JMP  L24C     IMPLIED RETURN

0785*
0786* STORE BLANKS. A REGISTER HAS THE MINUS COUNT.
0787*
0788  02440 070313 L24A   SAP  L24E     IF A NOT < 0 THEN RET
0789  02441 031723       STA  N        SAVE COUNT
0790  02442 020360       LDA  DP32     STORE ASCII
0791  02443 062001       JSM  .BF3I-1    BLANK IN THE BUFFER
0792  02444 045723       ISZ  N        MORE BLANKS NEEDED?
0793  02445 066442       JMP  *-3        YES, CONTINUE TO LOOP
0794  02446 170402 L24E   RET           NO, RETURN
0795*
0796* STORE SIGN OF MANTISSA OR EXPONENT AS FOLLOWS.
0797*
0798*     E=0  MANTICSA
0799*     E=1  EXPONENT
0800*
0801  02447 021754 L24B   LDA  AR2      GET EXPONENT WORD WITH THE SIGN
                                           INFORMATION
0802  02450 070114       SES  *+2      IF E=0
0803  02451 070006       RAR  1          THEN SET A(15)←SIGN OF MANTISSA
0804  02452 070153       SAP  *+3      IF A(15)=1
0805  02453 020527       LDA  DP45       THEN STORE ASCII MINUS SIGN IN THE
                                           BUFFER
0806  02454 066001       JMP  .BF3I-1  IMPLIED RETURN
0807*
0808  02455 020360       LDA  DP32     STORE BLANK IN THE BUFFER
0809  02456 066001       JMP  .BF3I-1  IMPLIED RETURN
0810*
0811* STORE DIGITS. A REGISTER SET TO NUMBER OF DIGITS TO BE STORED.
0812* BEFORE EACH DIGIT IS STORED, A CHECK IS MADE TO SEE
```

```
0813*  THAT I/O BUFFER IS NOT FULL.
0814*
0815   02457  031721  L24C   STA  L        SAVE THE COUNT
0816   02460  021706         LDA  .POS     IF THE BUFFER IS FULL
0817   02461  071250         SZA  L24E       THEN EXIT THIS ROUTINE
0818   02462  171400         MLS           GET NEXT DIGIT
0819   02463  000355         ADA  DP48     FORM ASCII CHARACTER BY ADDING THE
                                              OFFSET
0820   02464  062001         JSM  .BF3-1   STORE THE CHARACTER IN THE BUFFER
0821   02465  055721         DSZ  L        MORE DIGITS?
0822   02466  066460         JMP  L24C+1     YES CONTINUE TO LOOP
0823   02467  170402         RET             NO, RETURN
0824*
0825*  .RND IS USED TO ROUND DIGIT N OF A DATA WORD STORED IN STANDARD
0826*  INTERNAL MACHINE FORM. THE GUARD DIGITS ARE ALSO ROUNDED.
0827*  THE CALLING SEQUENCE IS:
0828*
0829*      LDA ADDRESS OF DATA WORD TO BE ROUNDED.
0830*      LDR DIGIT POSITION TO BE ROUNDED
0831*      JSM .RND
0832*
0833*  IF MANTISSA OVERFLOW OCCURES, BOTH THE MANTISSA AND EXPONENT
0834*  WILL BE ADJUSTED AND THE A REGISTER SET TO -1.
0835*
0836   02470  024367         LDB  DP13     USED INEQUALITIES
0837   02471  035730  .RND   STB  I1       SAVE DIGIT POSITION
0838   02472  075652         SBM  *-3      IF B NOT > 0
0839   02473  075610         SZB  *-4        THEN RETURN
0840   02474  024450         LDB  DAR2     TRANSFER DATA WORD TO BE
0841   02475  170004         XFR             ROUNDED INTO AR2
0842   02476  020447         LDA  DAR1     CLEAR AR1
0843   02477  170000         CLR
0844   02500  020377         LDA  DP5      SET A 5 INTO DIGIT POSITION
0845   02501  025730         LDB  I1         OF AR1 THAT IS TO BE ROUNDED
0846   02502  174430         MRX
0847*
0848*  ALSO ROUND DIGIT 11.
0849*
0850   02503  020347         LDA  DP80     INCLUSINVE OR A 5 INTO DIGIT
0851   02504  041747         IOR  AR1+3      11 (LEADING GUARD DIGIT) OF AR1
0852   02505  031747         STA  AR1+3
0853   02506  070075         SEC  *+1,C    SET E←0
0854   02507  170560         FXA           DO A FLOATING ADD
0855*
0856*  TEST FOR MANTISSA OVERFLOW.
0857*
0858   02510  021757         LDA  AR2+3
0859   02511  050427         AND  DM256    ZERO D11 AND D12
0860   02512  031757         STA  AR2+3
0861   02513  170002         MOV           A←E
0862   02514  070350         SZA  L24F     IF A=0 THE RETURN
0863   02515  070137         LDB  A        MOVE THE CARRY DIGIT INTO MANTISSA FIELD
0864   02516  174470         MRY
0865   02517  025754         LDB  AR2      INCREMENT EXPONENT VALUE
0866   02520  004341         ADB  DP256
0867   02521  035754         STB  AR2
0868   02522  024405         LDB  DM1
```

I/O SUPERVISOR

```
0869  02523  020450 L24F   LDA DAR2
0870  02524  170402        RET
0871*
0872* .BXFR TRANSFERS THE CONTENTS OF BUFFER <A> TO BUFFER <B>.
0873* ALL TRANSFERS ARE 8 WORDS IN LENGTH
0874*
0875  02525  031721 .BXFR STA L       SAVE L←A
0876  02526  035720        STB K         AND K←B
0877  02527  024413        LDB DM7     SET B←-7
0878  02530  121721        LDA L,I     TRANSFER ALL 8 WORDS
0879  02531  131720        STA K,I       FROM BUFFER L TO BUFFER K
0880  02532  045721        ISZ L
0881  02533  045720        ISZ K
0882  02534  077630        RIB *-4     IF NOT DONE THEN CONTINUE TO LOOP
0883  02535  170402        RET         RETURN
```

(96)

I/O SUPERVISOR

PRINTER ROUTINES

```
0885*
0886************************************
0887*
0888*
0889*  THE BIT PATTERN FOR THE THERMO PRINTER IS AS FOLLOWS:
0890*
0891*     DDDDDCCCCCXXXSGG    2ND WORD
0892*
0893*     BBBBBAAAAAXXXXXX    1ST WORD
0894*
0895*  WHERE A = 1ST CHARACTER
0896*        B = 2ND CHARACTER
0897*        C = 3RD CHARACTER
0898*        D = 4TH CHARACTER
0899*        G = 1 OF 4 GROUP CODE
0900*        S = ROW ADVANCE
0901*        X = NOT USED
0902*
0903************************************
0904*
0905*  SET ROW COUNT, SHIFT COUNT, AND MASK.
0906*
0907  02536  020413  .PRT1  LDA  DM7     SET ROW COUNT TO -7
0908  02537  031727         STA  T         AND SAVE IT IN T
0909  02540  020434         LDA  K5      SET "SBR" 7" INSTRUCTION
0910  02541  031726         STA  S         AND SAVE IT IN S
0911  02542  020444         LDA  K19     SET BIT MASK
0912  02543  031725         STA  R         AND SAVE IT IN R
0913*
0914*  INITIALIZE I1, I2, and I3.
0915*
0916  02544  062662         JSM  L3F     INITIALIZE I1, I2, and I3
0917  02545  066554         JMP  L3A     CONTINUE AT L3A
0918*
0919*  ADVANCE PRINTER ONE ROW THEN
0920*  SET CHARACTER COUNT, BUFFER ADDRESS, MASK,
0921*  AND MODIFY SHIFT COUNT
0922*
0923  02546  021726  L3     LDA  S       DECREMENT SHIFT
0924  02547  000423         ADA  DM32      INSTRUCTION BY 1
0925  02550  031726         STA  S           AND SAVE IT
0926  02551  025725         LDB  R       SHIFT BIT MASK RIGHT
0927  02552  074002         SBR  1         BY ONE
0928  02553  035725         STB  R           AND SAVE IT
0929  02554  020420  L3A    LDA  DM16    SET CHARACTER COUNT<--16
0930  02555  031720         STA  K         AND SAVE IT IN K
0931  02556  020207         LDA  .DBUF   GET ADDRESS OF CHARACTER BUFFER
0932  02557  031721         STA  L         AND SAVE IT IN L
0933*
0934*  DO SUBROUTINE AND CLEAR P REGISTER.
0935*
0936  02560  062672  L3C    JSM  L3B+1   GET PATTERN INFORMATION FOR NEXT
                                            CHARACTER
0937  02561  025725         LDB  R       B<-BIT MASK
0938  02562  070742         SAR  16      A<-0
0939*
0940*  FORM BIT PATTERN
```

```
0941*
0942   02563  031724 LI    STA   P      SAVE LAST RESULT
0943   02564  121722       LDA   M,I    GET 7 BIT PATTERN
0944   02565  074257       AND   B      MASK PROPER BIT
0945   02566  041724       IOR   P      INCLUDE LAST PATTERN RESULT
0946   02567  070706       RAR   15     POSITION RESULT LEFT 1
0947   02570  045722       ISZ   M      INCREMENT ADDRESS TO GET NEXT
                                          PATTERN
0948   02571  045723       ISZ   N      HAVE ALL 5 BITS BEEN GATHERED?
0949   02572  066563       JMP   L1       NO, CONTINUE TO LOOP
0950*
0951*  POSITION AND MASK 5 BITS.
0952*
0953   02573  070137       LDB   A      SET B←A
0954   02574  070114       SES   *+2    PATTERN IN HIGH PART OF A?
0955   02575  074346       RBR   8        YES, POSITION IT LOW
0956   02576  021726       LDA   S      SET A←VARIABLE SHIFT INSTRUCTION
0957   02577  070016       EXA          EXECUTE THE INSTRUCTION FOUND IN A
0958   02600  074117       LDA   B      A←B
0959   02601  050361       AND   DP31   SAVE ONLY LAST 5 BITS
0960*
0961*  INCLUDE LAST 5 BITS AND CHECK FOR END OF FIRST
0962*  WORD PROCESSED. IF DONE STORE IN 14 and RESET 13.
0963*
0964   02602  041732       IOR   I3     INCLUDE LAST 5 BIT PATTERN (13=0
                                          OTHERWISE)
0965   02603  070206       RAR   5      POSITION NEW RESULT
0966   02604  045731       ISZ   I2     FIRST WORD FILLED?
0967   02605  066610       JMP   *+3      NO, SKIP 2 INSTRUCTIONS
0968   02606  031733       STA   I4       YES, SAVE 14←A
0969   02607  070742       SAR   16         AND SET A←0
0970*
0971*  CONTINUE PROCESSING. STORE A IN 13 AND TEST TO SEE IF
0972*  GROUP IS READY FOR OUTPUT.
0973*
0974   02610  031732       STA   I3     SAVE 13←A
0975   02611  045730       ISZ   I1     SECOND WORD FILLED?
0976   02612  066622       JMP   L1A      NO, CONTINUE AT L1A
0977*
0978*  INCLUDE GROUP COUNT AND OUTPUT PATTERN. THEN
0979*  INITIALIZE I1, I2, AND I3.
0980*
0981   02613  021720       LDA   K      COMPUTE NUMBER OF CHARACTERS
0982   02614  000364       ADA   DP16     PROCESSED SO FAR
0983   02615  070042       SAR   2      DIVIDE BY 4 TO GET GROUP CODE
0984   02616  041732       IOR   I3     INCLUDE SECOND WORD 5 BIT PATTERNS
0985   02617  025733       LDB   I4     LOAD B←FIRST WORD 5 BIT PATTERNS
0986   02620  062642       JSM   L3E    PRINTER OUTPUT SUBROUTINE
0987   02621  062662       JSM   L3F    RESET I1, I2, AND I3
0988*
0989*  INCREMENT CHARACTER COUNT.
0990*
0991   02622  045720 L1A   ISZ   K      ALL 16 CHARACTERS PROCESSED?
0992   02623  066560       JMP   L3C      NO, LOOP TO PROCESS NEW CHARACTER
                                          AT L3C
0993*
0994*  CHECK ROW COUNT. IF DONE ADVANCE PRINTER 3 ROWS, TURN ON
0995*  THE INTERRUPT SYSTEM, AND RETURN.
0996*
```

I/O SUPERVISOR

```
0997  02624 045727        ISZ  T       ALL 7 ROWS FINISHED?
0998  02625 066546        JMP  L3        NO, LOOP TO PROCESS NEW LINE AT L3
0999  02626 020401        LDA  DP3     SET A←3
1000  02627 070076 .SPAC  TCA          SET←—A
1001  02630 031727        STA  T         AND SAVE IN T
1002  02631 062640        JSM  L3D     ADVANCE PRINTER 1 ROW
1003  02632 045727        ISZ  T       DONE?
1004  02633 066631        JMP  *-2       NO, CONTINUE TO LOOP
1005  02634 172704        SFC  4       WAIT FOR PRINT OPERATION TO FINISH
1006  02635 066634        JMP  *-1
1007  02636 173741        CLF  1       TURN ON THE INTERRUPT SYSTEM
1008  02637 170402        RET          RETURN
1009*
1010* L3D SETS UP THE A AND B REGISTERS TO ADVANCE THE PRINTER
1011* ONE ROW.
1012*
1013  02640 020401 L3D     LDA  DP3     A←GROUP 4 CODE AND NON-PRINTING PATTERN
1014  02641 074742         SBR  16      B←NON-PRINTING PATTERN
1015*
1016* L3E CHECKS THE PRINTER FLAG AND OUTPUTS THE A AND B REGISTERS.
1017*
1018  02642 031723 L3E     STA  N       SAVE N←A
1019  02643 020425         LDA  K16     SET A←TIME COUNT
1020  02644 070470         SIA  L3G     IF A=0 THEN GO TO L3G ELSE A←A+1
1021  02645 172704         SFC  4       IF PRINTER FLAG SET
1022  02646 066644         JMP  *-2       THEN CONTINUE TO LOOP
1023  02647 021723         LDA  N       ELSE RESTORE A←N
1024  02650 173741         CLF  1       TURN ON THE INTERRUPT SYSTEM
1025  02651 172741         STF  1       TURN OFF THE INTERRUPT SYSTEM
1026  02652 176141         OTB  1       OUTPUT 1ST WORD
1027  02653 172144         OTA  4       OUTPUT 2ND WORD. IMPLIED STC 4
1028  02654 170402         RET          RETURN
1029*
1030* PRINTER OUT OF PAPER. SET ERROR MESSAGE AND WAIT FOR STOP
1031* OR EXEC KEYCODE.
1032*
1033  02655 024364 L3G     LDB  DP16    SET A←ERROR 16
1034  02656 055777         DSZ  1777B   DECREMENT JSM STACK
1035  02657 063251         JSM  L7      DISPLAY ERROR MESSAGE AND WAIT FOR
                                          KEYCODE
1036  02660 170402         RET          STOP KEY. EXIT PRINT ROUTINE
1037  02661 066536         JMP  .PRT1   EXECUTE KEY. RE-ENTER PRINT ROUTINE
1038*
1039* L3F SETS UP I1, I2, AND I3.
1040*
1041  02662 020410 L3F     LDA  DM4     SET I1←—4
1042  02663 031730         STA  I1
1043  02664 020406         LDA  DM2     I2←—2
1044  02665 031731         STA  I2
1045  02666 070742         SAR  16      I3←0
1046  02667 031732         STA  I3
1047  02670 170402         RET          RETURN
```

I/O SUPERVISOR

```
1049*
1050*  GET BIT PATTERN ADDRESS, SET LOOP COUNTER, AND
1051*  INCREMENT BUFFER ADDRESS AS NEEDED.
1052*
1053  02671  173750  L3B   CLF  8          CLEAR DISPLAY FLAG
1054  02672  025720        LDB  K          B←NEGATIVE CHARACTER COUNT
1055  02673  121721        LDA  L,I        GET CHARACTER BUFFER WORD SPECIFIED BY L
1056  02674  074151        SLB  *+3        CHARACTER IN HIGH PART OF WORD?
1057  02675  070346        RAR  8          NO, POSITION IT HIGH
1058  02676  045721        ISZ  L            AND INCREMENT BUFFER ADDRESS
                                             POINTER
1059  02677  070346        RAR  8          POSITION CHARACTER FROM HIGH TO LOW
                                             PART OF A
1060  02700  050345        AND  DP127      SAVE ONLY THE LAST 7 BITS OF A
1061  02701  070075        SEC  *+1,C      SET E←0
1062  02702  070131        SLA  *+2,C      IF A(0)=1 (ALSO SET A(0)←0 IN ANY CASE)
1063  02703  072054        SES  *+1,S        THEN SET E←1
1064  02704  070137        LDB  A          COMPUTE A←5*A/2
1065  02705  074002        SBR  1            TO FORM RELATIVE ADDRESS
1066  02706  070706        RAR  15             OF THE PATTERN ARRAY
1067  02707  070037        ADB  A
1068  02710  004245        ADB  .TAB5      ADD PATTERN ARRAY ADDRESS
1069  02711  035722        STB  M            AND SAVE M←A
1070  02712  024411        LDB  DM5        SET N←-5
1071  02713  035723        STB  N
1072  02714  170402        RET             RETURN
```

```
1074*
1075*****************************************************
1076*
1077*   THE FOLLOWING IS THE COMPLETE DRIVER FOR THE DISPLAY. IT
1078*   WILL PERFORM THE FOLLOWING FUNCTIONS.
1079*
1080*   1.  CHECK FOR KEYCODES.  WHEN A KEYCODE IS ENTERED,
1081*       .WMOD IS RESET AND CONTROL WILL RETURN TO THE
1082*       CALLING PROGRAM WITH THE ENTERED KEYCODE IN THE
1083*       A REGISTER.
1084*
1085*   2.  DRIVE THE DISPLAY AS FOLLOWS
1086*
1087*       A.  THE CONTENTS OF THE DISPLAY BUFFER IF NO
1088*           ERROR HAS OCCURED.
1089*
1090*       B.  ALTERNATE BETWEEN THE ERROR MESSAGE AND THE
1091*           CONTENTS OF THE DISPLAY REGISTER IF AN ERROR
1092*           HAS OCCURED.
1093*
1094*****************************************************
1095*
1096*   ON ENTRY CHECK FOR ERROR.
1097*
1098    02715 062740 .DSP  JSM   L2B       CHECK FOR ERROR
1099    02716 066723       JMP   L2A
1100*
1101*   DO A DISPLAY CYCLE AFTER FIRST DETERMINING WHETHER TO USE
1102*   THE DISPLAY OR I/O REGISTER.
1103*
1104    02717 021704 L2E   LDA   .WBUF
1105    02720 031721       STA   L         SAVE ADDRESS OF THE BUFFER TO BE
                                             DISPLAYED
1106    02721 063000       JSM   .DSP1     DISPLAY DRIVER SUBROUTINE
1107    02722 066726       JMP   L2A+3
1108*
1109*
1110*   CHECK FOR KEYCODE ENTERED.  IF ENTERED, RESET .WMOD, .IBUF, AND
1111*   EXIT WITH KEYCODE IN A REGISTER.
1112*
1113    02723 021704 L2A   LDA   .WBUF    NO ERROR SET UP TO DISPLAY .DBUF
1114    02724 052777       AND   DM8
1115    02725 031704       STA   .WBUF
1116    02726 021705       LDA   .WMOD    IF .WMOD(15)=0 THEN CONTINUE
1117    02727 071433       SAP   L2E,C      TO DIPLAY AT L2E
1118    02730 031705       STA   .WMOD      FLAG FROM .WMOD
1119    02731 020554 L2F   LDA   .IBUF    RESET .IBUF
1120    02732 062227       JSM   L18A
1121    02733 020340       LDA   BGNLE    STORE A BEGINNING OF LINE
1122    02734 131704       STA   .WBUF,I    CHARACTER INTO BUFFER SPECIFIED BY
                                             .WBUF
1123    02735 021701       LDA   .WKC     A<-KEYCODE
1124    02736 050342       AND   DP255    SAVE ONLY BITS 0-7
1125    02737 170402 RET   RET
1126*
1127*   SET-UP ERROR MESSAGE INTO I/O REGISTER.
1128*
1129*   02740 021705 L2B   LDA   .WMOD
```

```
1130  02741  070102          SAR  3
1131  02742  050342          AND  DP255     GET ERROR NUMBER
1132  02743  071610          SZA  RET       IF ZERO THEN RETURN — NO ERROR
1133  02744  031722          STA  M
1134  02745  022757          LDA  L2D       A←ADDRESS ASCII MESSAGE TO BE
                                              PLACED IN THE DISPLAY
1135  02746  024554          LDB  .IBUF     B←ADDRESS OF THE I/O BUFFER
1136  02747  062525          JSM  .BXFR     DO AN 8 WORD TRANSFER
1137  02750  020554          LDA  .IBUF     FORM ADDRESS OF THE 8TH
1138  02751  000402          ADA  DP2         WORD OF THE I/O BUFFER
1139  02752  031704          STA  .WBUF     SAVE .WBUF←A
1140  02753  020461          LDA  K110
1141  02754  031706          STA  .POS      .POS←HI/LO POSITION WORD (52525B)
1142  02755  021722          LDA  M         A←ERROR CODE
1143  02756  066340          JMP  L24G      DO ERROR NUMBER CONVERSION AND
                                              STORAGE
1144*
1145*  ERROR MESSAGE
1146*
1147  02757  002760  L2D      DEF  *+1       ADDRESS OF ERROR MESSAGE
1148  02760  047117           ASC  8,NOTE
1149  02770  044516  IN       ASC  1,IN
1150*
1151*  RUN TIME ERROR MESSAGE
1152*
1153  02771  062740  RNTER    JSM  L2B
1154  02772  045704           ISZ  .WBUF
1155  02773  022770           LDA  IN
1156  02774  131704           STA  .WBUF,I
1157  02775  045704           ISZ  .WBUF
1158  02776  164523           JMP  STLN,I
1159*
1160  02777  177770  DM8      DEC  -8
```

I/O SUPERVISOR

```
1162*
1163************************************
1164*
1165*  THE FOLLOWING IS THE DISPLAY SIO DRIVER FOR THE SYSTEM.
1166*
1167*  THE BIT PATTERN FOR THE LED IS:
1168*
1169*      XXEEEFFFFBBBBBBB
1170*  WHERE B IS THE COLUMN BIT PATTERN
1171*        E IS THE 1 OF 8 CODE ADDRESS
1172*        F IS THE BCD ADDRESS
1173*        X ARE THE UNUSED BITS.
1174*
1175****************************************
1176*
1177*  ON ENTRY, L MUST HAVE ALREADY BEEN SET TO THE ADDRESS OF THE
1178*  BUFFER TO BE OUTPUT.
1179*
1180  03000  020420  .DSP1 LDA  DM16   SET K←-16
1181  03001  031720        STA  K      (NEGATIVE CHARACTER COUNT)
1182*
1183*  SET BCD COUNTER & ADDRESS WORD.
1184*
1185  03002  020415        LDA  DM10   R←-10
1186  03003  031725        STA  R      (NEGATIVE BCD COUNT)
1187  03004  070742        SAR  16     S←0
1188  03005  031726        STA  S      (LED ADDRESS INFORMATION)
1189*
1190*  START PROCESSING NEW CHARACTER.
1191*
1192  03006  062671  L4D   JSM  L3B    GET PATTERN INFORMATION FOR NEXT
                                       CHARACTER
1193  03007  121722        LDA  M,I    GET 7 BIT PATTERN
1194  03010  070114        SES  *+2    PATTERN IN HIGH PART OF WORD?
1195  03011  070346        RAR  8        YES, POSITION IT TO LOW PART OF A
1196  03012  050345        AND  DP127  SAVE ONLY LAST 7 BITS
1197  03013  041726        IOR  S      INCLUDE LED ADDRESS INFORMATION
1198  03014  173750        CLF  8      CLEAR DISPLAY FLAG
1199  03015  173741        CLF  1      TURN OF THE INTERRUPT SYSTEM
1200  03016  172741        STF  1      TURN OFF THE INTERRUPT SYSTEM
1201  03017  172150        OTA  8      OUTPUT DISPLAY INFORMATION. IMPLIED
                                       STC 8
1202*
1203*  CHANGE ADDRESS WORD. INCREMENT BCD COUNT AND IF 0, DO SECOND
1204*  CHANGE ON ADDRESS WORD AND RESET BCD COUNT.
1205*
1206  03020  021726        LDA  S      INCREMENT BCD ADDRESS PART
1207  03021  000344        ADA  DP128    OF WORD S
1208  03022  045725        ISZ  R      BCD OVERFLOW?
1209  03023  067030        JMP  *+5      NO, SKIP 4 INSTRUCTIONS
1210  03024  050445        AND  K20    SAVE AND INCREMENT 1 OF 8
1211  03025  000446        ADA  K21      CODE PART OF S
1212  03026  024415        LDB  DM10   RESET R←-10
1213  03027  035725        STB  R
1214  03030  031726        STA  S      SAVE S←A
1215*
1216*  INCREMENT BUFFER ADDRESS AND CHARACTER COLUMN COUNT. IF DONE
1217*  CHECK CHARACTER COUNT, ELSE FINISH PRESENT CHARACTER.
```

I/O SUPERVISOR

```
1218*
1219   Ø3Ø31  Ø45722        ISZ   M       INCREMENT PATTERN ARRAY ADDRESS
                                             POINTER
1220   Ø3Ø32  Ø45723        ISZ   N       ALL 5 COLUMNS DONE?
1221   Ø3Ø33  Ø67ØØ7        JMP   L4D+1     NO, CONTINUE TO LOOP AT L4D+1
1222*
1223*  INCREMENT CHARACTER COUNT. IF DONE THEN EXIT, ELSE START
1224*  NEW CHARACTER.
1225*
1226   Ø3Ø34  Ø4572Ø        ISZ   K       ALL 16 CHARACTERS DONE?
1227   Ø3Ø35  Ø67ØØ6        JMP   L4D       NO, CONTINUE TO LOOP AT L4D
1228*
1229*  TURN OFF DISPLAY
1230*
1231   Ø3Ø36  Ø2Ø41Ø        LDA   DM4     DELAY FOR LAST
1232   Ø3Ø37  Ø72Ø3Ø        RIA   *         COLUMN THAT WAS OUTPUT
1233   Ø3Ø4Ø  173750        CLF   8       TURN OFF DISPLAY
1234   Ø3Ø41  173741        CLF   1       TURN ON THE INTERRUPT SYSTEM
1235   Ø3Ø42  17Ø4Ø2        RET           RETURN
1236*
```

(104)

MAGNETIC CARD ROUTINES

LOAD AND RECORD ROUTINES FOR MAGNETIC CARD

```
1238*
1239*  .REC IS USED FOR RECORDING PROGRAM AND/OR DATA INFORMATION
1240*  ON MAGNETIC CARDS IN EITHER A SECURE OR UNSECURE MODE.
1241*
1242*  FIRST SET UP NECESSARY PARAMETERS AND CHECK TO SEE IF SECURE
1243*  OR DATA MODE HAS BEEN SET.
1244*
1245  03043 063146 .REC JSM  L5S     SET LINK WORDS FOR RECORD
1246  03044 070410      SZA  L5D     IF NO PARAMETERS RCORD ONLY PROGRAM
1247  03045 074517      LDA  B,I     B←ADDRESS OF FIRST PARAMETER
1248  03046 050515      AND  K22     SAVE BITS 0–13
1249  03047 070517      LDA  A,I     GET FIRST WORD OF FIRST PARAMETER
1250  03050 010463      CPA  K8      IF A="SE" (SECURE MODE)
1251  03051 067143      JMP  L5H       THEN GO TO L5H
1252  03052 010443      CPA  K1      IF A="D" (DATA ONLY)
1253  03053 067100      JMP  L5R
1254  03054 063104 L5D  JSM  L5P     RECORD PROGRAM
1255  03055 045554      ISZ  PCNT
1256  03056 067101      JMP  L5R+1   CHECK PARAMETER COUNT FOR DATA
1257*
1258*  SET ADDRESS OF NEXT DATA WORD.  IF TEMP. OR ALPHA INFORMATION,
1259*  SET ERROR 2 AND RETURN TO SYSTEM.
1260*
1261  03057 125555 L5E  LDB  PADD,I  A←ADDRESS OF NEXT DATA WORD
1262  03060 074117      LDA  B       B←A
1263  03061 070642      SAR  14      IF A(15–14)=0 (REGISTER REFERENCE)
1264  03062 070150      SZA  *+3       THEN SKIP 2 INSTRUCTIONS
1265  03063 024402      LDB  DP2       ELSE B←ERROR NUMBER 2
1266  03064 067141      JMP  L5F+1     DISPLAY ERROR NUMBER
1267*
1268*  IS ADDRESS OUTSIDE OF SYSTEM MEMORY?  IF YES, SET RECORD LENGTH
1269*  FOR R REGISTERS FROM MAW TO ADDRESS SPECIFIED.  ELSE SET LENGTH
1270*  TO –4.  IN EITHER CASE, LENGTH IS NEGITIVE.
1271*
1272  03065 020441      LDA  K13     A←1777B — TOP END OF SYSTEM R/W
                                       MEMORY
1273  03066 070076      TCA          A←–A
1274  03067 074017      ADA  B       ADD ADDRESS OF DATA WORD
1275  03070 070153      SAP  *+3     IF A,B,C,X,Y, OR Z REFERENCE
1276  03071 020410      LDA  DM4     THEN A←–4 (4 WORDS IN THE DATA MODE)
1277  03072 067077      JMP  *+5     SKIP 4 INSTRUCTIONS
1278  03073 021377      LDA  MAW     A←LAST WORD OF MEMORY+1
1279  03074 070076      TCA          A←–A
1280  03075 074017      ADA  B       ADD ADDRESS OF DATA WORD TO GET
                                       LENGTH
1281  03076 076052      SBM  *+1,S   B(15)←1 TO INDICATE RELATIVE ADDRESS
                                       STRUCTURE
1282  03077 063120      JSM  L5B     RECORD DATA
1283*
1284*  CHECK PARAMETER COUNT AND IF DONE RETURN TO SYSTEM.
1285*
1286  03100 055555 L5R  DSZ  PADD    DECREMENT POINTER TO ADDRESS OF
                                       NEXT DATA WORD
1287  03101 055554      DSZ  PCNT    MORE PARAMETERS?
1288  03102 067057      JMP  L5E       YES, PROCESS AT L5E
1289  03103 165755      JMP  .SYM,I    NO, TURN OFF MAG CARD
1290*
1291*  LINK WORD SETUP FOR MAG-CARD
1292*  LOAD AND RECORD
1293*
```

LOAD AND RECORD ROUTINES FOR MAGNETIC CARD

```
1294*  RECORD CONFIGURATION CODE AND SECURITY WORD.
1295*
1296   03104  021625 L5P   LDA  .SE      GET SECURE WORD
1297   03105  070110       SZA  *+2      IF ZERO ALLOW RECORDING OF CURRENT
                                          PROGRAM
1298   03106  066136       JMP  .INT2    ELSE RESET SYSTEM
1299*
1300   03107  020410 L5O   LDA  DM4      LOAD A — REG WITH WORD COUNT
1301   03110  024447       LDB  DAR1     LOAD B — REG WITH ADDRESS OF DATA
1302   03111  161754       JSM  .REC,I     RECORD CONFIGURATION CODE
1303   03112  070110       SZA  *+2
1304   03113  067122       JMP  L5B+2
1305*
1306*  RECORD PROGRAM INFORMATION STARTING AT PRESENT LOCATION OF
1307*  PROGRAM COUNTER (CMA) TO ENDS.  IF CMA=ENDS THEN SKIP PROGRAM
1308*  RECORDING.
1309*
1310   03114  021546       LDA  CMA
1311   03115  070076       TCA
1312   03116  001542       ADA  ENDS
1313   03117  025546       LDB  CMA
1314*
1315*  L5B IS USED TO CALL THE RECORD DRIVER AND CHECK FOR ERROR
1316*  CODES ON RETURN.  THE CODES ARE:
1317*
1318*       A=0  NORMAL
1319*            1  SHORT CARD                          ERROR 14 — CALL .DSP
1320*            2  CARD PROTECTED                      ERROR 13 — RETURN
1321*            3  CARD NOT IN AND STOP PRESSED        NO ERROR — RETURN
1322*
1323   03120  161754 L5B   JSM  .REC,I   CALL MAG CARD RECORD SUBROUTINE
1324   03121  071110       SZA  L5P-1    IF A=0, NO ERROR SO RETURN
1325   03122  000406       ADA  DM2      ELSE LOOK FOR CAUSE OF STOP
1326   03123  070553       SAP  L5C      IF — SHORT CARD
1327*
1328*  DISPLAY ERROR MESSAGE 14 AND WAIT FOR STOP OR EXEC
1329*  KEYCODE.
1330*
1331   03124  161755 L5T   JSM  .SYM,I
1332   03125  024366       LDB  DP14
1333   03126  063251       JSM  L7       DISPLAY ERROR 14 AND WAIT FOR KEYCODE
1334   03127  067136       JMP  L5C      STOP KEY.  GO TO L5C
1335   03130  024405       LDB  DM1      EXECUTE KEY.
1336   03131  035727       STB  T        SET T←—1
1337   03132  025730       LDB  TEMP     B←CONTINUATION ADDRESS
1338   03133  021731       LDA  WDCNT    A←WORD COUNT
1339   03134  071212       SAM  L5B      IF COUNT — DATA CONTINUE RECORDING
1340   03135  067107       JMP  L5O        ELSE PROGRAM: PUT CONFIGURATION
                                          ON NEXT PASS
1341*
1342   03136  055777 L5C   DSZ  1777B
1343   03137  072150       RZA  L5F+2    IF NOT ZERO STOP PRESSED
1344   03140  024367 L5F   LDB  DP13
1345   03141  061351       JSM  BAD
1346   03142  165755       JMP  .SYM,I

1348*
```

(106)

LOAD AND RECORD ROUTINES FOR MAGNETIC CARD

```
1349*  SECURITY RECORDING.  SET SECURITY WORD, ADJUST PARAMETERS, AND
1350   DO RECORDING OF PROGRAM AND/OR DATA.
1351*
1352   03143  055744  L5H   DSZ   AR1      SET SECURITY WORD
1353   03144  063104        JSH   L5P      RECORD PROGRAM
1354   03145  067100        JMP   L5R      CHECK FOR MORE PARAMETERS

1356*
1357*  RECORD AND LOAD SET UP.
1358*
1359*  SETUP LINK ADDRESSES TO DRIVERS OF THE MAG-CARD
1360*  GET PARAMETER COUNT, PARAMETER ADDRESS, AND TRANSFER
1361*    CONFIGURATION CODE.
1362*
1363   03146  023605  L5S   LDA   RLINK    LINK WORD TO MAG-CARD RECORD
1364   03147  031754        STA   .REC.
1365   03150  023606        LDA   RLINK+1  LINK WORD TO MAG-CARD SHUTDOWN
1366   03151  031755        STA   .SYM
1367   03152  023607        LDA   RLINK+2  LINK WORD TO MAG-CARD LOAD
1368   03153  031756        STA   .LOD.
1369*
1370   03154  020405  L5A   LDA   DM1
1371   03155  031727        STA   T
1372   03156  000247        ADA   .TAB7    A←ADDRESS OF CONFIGURATION CODE
                                            TABLE
1373   03157  024447        LDB   DAR1     B←ADDRESS OF AR1
1374   03160  170004        XFR            DO A 4 WORD TRANSFER
1375*
1376   03161  021556        LDA   AP1
1377   03162  070076        TCA
1378   03163  001557        ADA   AP2      COMPUTE # PARAMETERS
1379   03164  031554        STA   PCNT     SAVE COUNT
1380   03165  025557        LDB   AP2      SET I←ADDRESS OF 1ST PARAMETER
                                            ADDRESS
1381   03166  004405        ADB   DM1
1382   03167  035555        STB   PADD
1383   03170  170402        RET            RETURN
```

LOAD AND RECORD ROUTINES FOR MAGNETIC CARD.

```
1385*
1386*  .LOD WILL LOAD A MAGNETIC CARD INTO THE 9820A MEMORY AND CHECK
1387*  FOR PROPER CONFIGURATION CODE.
1388*
1389*
1390   03171  063146  .LOD  JSM  L5S
1391   03172  021563        LDA  TSW      IF TSW NOT<0
1392   03173  070152        SAM  *+3
1393   03174  021604        LDA  NMA        OR NMA=0
1394   03175  070110        SZA  *+2        THEN PREVENT LOADING
1395   03176  067205        JMP  L5G
1396   03177  024370        LDB  DP12
1397   03200  061351        JSM  BAD
1398   03201  164346        JMP  XJ,I
1399*
1400*  CALL DRIVER TO LOAD FROM CARD AND CHECK FOR THE FOLLOWING
1401*  ERRORS ON RETURN.
1402*
1403*    A=0  NORMAL                    NO ERROR — CONT.
1404*      1  STOP                       NO ERROR — RETURN
1405*      2  SHORT CARD                 ERROR 14 — CALL .DSP
1406*      3  MEMORY OVERFLOW            ERROR 12 — RETURN
1407*
1408   03202  024366  L5J   LDB  DP14
1409   03203  063251        JSM  L7
1410   03204  067216        JMP  L5K      STOP RETURN
1411*
1412   03205  161756  L5G   JSM  .LOD.,I  CALL MAG CARD LOAD SUBROUTINE
1413   03206  070137        LDB  A
1414   03207  161755        JSM  .SYM,I
1415   03210  004406        ADB  DM2      LOOK FOR WHY STOP?
1416   03211  074252        SBM  L5K      STOP KEY OR NORMAL RETURN
1417   03212  075410        SZB  L5J        SHORT CARD
1418   03213  024370  L5N   LDB  DP12       ELSE MEMORY OVERFLOW OCCURED
1419   03214  061351        JSM  BAD      SET ERROR 12
1420   03215  067237        JMP  L5L+2
1421*
1422*  STORE SECURITY WORD. IF NON-ZERO, SYSTEM CAN ONLY RUN PROGRAM.
1423*  CHECK CONFIGURATION CODE. IF CODE READ FROM THE CARD IS
1424*  A ZERO OR MATCHES THAT OF THE MACHINE, THEN SET ENDS TO
1425*  THE PRESENT VALUE OF CMA. ELSE SET ERROR 15.
1426*
1427   03216  020447  L5K   LDA  DAR1
1428   03217  031717        STA  J        AREA OF CARD LOADED CONFIGURATION
1429   03220  020247        LDA  .TAB7
1430   03221  031716        STA  I        AREA OF SYSTEM CONFIGURATION
1431*
1432   032222 021744        LDA  AR1      GET SECURE WORD FROM CARD
1433   03223  070110        SZA  *+2      IF ZERO DON'T CHANGE SYSTEM WORD
1434   03224  031625        STA  .SE
1435*
1436   03225  024406        LDB  DM2
1437   03226  045717  L5M   ISZ  J
1438   03227  121717        LDA  J,I      GET WORD FROM CARD CONFIGURATION
1439   03230  070250        SZA  L5L      IF WORD ZERO DON'T CHECK
1440   03231  111716        CPA  I,I      CHECK AGAINST SYSTEM
```

LOAD AND RECORD ROUTINES FOR MAGNETIC CARD.

```
1441   03232  067235          JMP  L5L     MATCH
1442*
1443   03233  024365          LDB  DP15    NO MATCH,ERROR #15
1444   03234  067214          JMP  L5N+1
1445*
1446   03235  045716  L5L     ISZ  I
1447   03236  077430          RIB  L5M
1448*
1449   03237  160320          JSM  RGTO,I  RESET ALL GTO'S AND GSB'S, A-REG HAS
                                             MAINLINE ADD.
1450   03240  055702          DSZ  GTS     WHAT TYPE LINK? GTO OR GSB
1451   03241  045702          ISZ  GTS     IF SKIP-GTO;THEN START WHERE CARD WAS
                                             LOADED
1452   03242  031604          STA  NMA     NOT SKIP-GSB;START AT BEGINNING OF
                                             MAIN LINE
1453   03243  160315          JSM  INTRP,I SETUP RUN TIME STACK
1454   03244  164316          JMP  XEOL,I
1455*
1456*
1457*  L7 IS THE ERROR MESSAGE ROUTINE THAT WILL CALL THE DISPLAY
1458*  ROUTINE AND WAIT FOR EITHER A STOP OR EXEC KEYCODE. THE
1459*  CALLING SEQUENCE IS AS FOLLOWS:
1460*
1461*       LDA ERROR CODE
1462*       JSM L7
1463*       ... STOP RETURN
1464*       ... NORMAL RETURN
1465*
1466   03245  061250          JSM  XSTP    SET STOP BIT
1467   03246  074742          SBR  16
1468   03247  061352          JSM  RBAD
1469   03250  067320          JMP  3320B   SET A-RER=3
1470   03251  061351  L7      JSM  BAD     SET ERROR USING I AD ROUTINE
1471   03252  020207          LDA  .DBUF
1472   03253  062732          JSM  L2F+1     CLEAR DISPLAY BUFFER
1473   03254  062715          JSM  .DSP    CALL DISPLAY MONITOR
1474   03255  010401          CPA  DP3
1475   03256  067245          JMP  L7-4    CLEAR KEY
1476   03257  010537          CPA  DP33    IF STOP KEY
1477   03260  067245          JMP  L7-4    THEN RETURN
1478   03261  010437          CPA  DP93    IF KEY=EXECUTE
1479   03262  067264          JMP  *+2       THEN SKIP 1 INSTRUCTION
1480   03263  067254          JMP  L7+3
1481*
1482   03264  063246          JSM  L7-3
1483   03265  025777  L7A     LDB  1777B   GET THE JSM STACK POINTER
1484   03266  004405          ADB  DM1     DECREMENT THE ADDRESS
1485   03267  074637          ISZ  B,I     INCREMENT THE RETURN ADDRESS
1486   03270  170402          RET
1487*
1488*********************************
1489*  ADD ON BLOCK REJECT ROUTINES FOR SYNTAX AND EXECUTION
1490*
1491   03271  021575  SYNTX   LDA  QUOTE   IF CODE IS IN QUOTE FIELD ALLOW IT
1492   03272  073710          RZA  *-2
1493   03273  024371  EXEQX   LDB  371B    ERROR #11
1494   03274  065351          JMP  BAD
1495                           END
*   NO ERRORS*
```

MAGNETIC CARD DRIVER ROUTINE

```
0001              ASMB,A,L
0002*
0003*   ALL OF THE FOLLOWING ARE OUTPUTS FROM THE 9820 TO
0004*   THE MAG CARD READER AND ARE POSITIVE TRUE LOGIC
0005*
0006*   BIT 0 (DO0) IS THE LSB OF THE DATA BYTE (CHANNEL A)
0007*   BIT 1 (DO1) IS THE NEXT LSB OF THE DATA BYTE (CHANNEL B)
0008*   BIT 2 (DO2) IS THE MSB OF THE DATA BYTE (CHANNEL C)
0009*   BIT 3 (DO3) IS THE STROBE DATA BIT
0010*   BIT 4 (DO4) IS THE INHIBIT CONTROL (0/1, OFF/ON)
0011*   BIT 5 (DO5) IS THE MOTOR CONTROL (0/1, OFF/ON)
0012*   STC 02 (MLS) IS THE 5 BIT LATCH CLOCK
0013*   STF 02 (MCR) IS THE STROBE BIT LATCH CLOCK
0014*
0015*   THE FOLLOWING ARE INPUTS TO THE 9820 FROM THE MAG
0016*   CARD READER AND ARE NEGATIVE TRUE POLARITY
0017*
0018*   BIT 0 (DI0) IS THE LSB OF THE DATA BYTE (CHANNEL A)
0019*   BIT 1 (DI1) IS THE NEXT LSB OF THE DATA BYTE (CHANNEL B)
0020*   BIT 2 (DI2) IS THE MSB OF THE DATA BYTE (CHANNEL C)
0021*   BIT 3 (DI3) IS THE END OF CARD SIGNAL WHEN RECORDING (SENSOR 1)
0022*   BIT 4 (DI4) IS THE CARD IN SIGNAL (SENSOR 2)
0023*   BIT 5 (DI5) IS THE PROTECTED CARD SIGNAL (SENSOR 3)
0024*   BIT 6 (DI6) IS THE END OF CARD SIGNAL WHEN LOADING (SENSOR 4)
0025*   THE DATA FLAG (MFL) IS THE DATA SAMPLING MARKER
0026*
0027*
0028*
0029*   IF THE LIGHT TO A SENSOR IS BLOCKED, THE OUTPUT LEVEL
0030*   IS GROUND OR A LOGICAL 1
0031  03306                ORG 3306B
0032  03306 070550 .REC.   SZA  *+11
0033  03307 031731         STA  WDCNT  A-REG ≠ WORDS. IF + PROGRAM:IF—DATA
0034  03310 035733         STB  ADMOD    B-REG ADDRESS OF A,B,C,X,Y,Z IF ADDRESS+
0035  03311 074072         SBM  FIX,C  IF ADDRESS (B-REG) NEGATIVE DATA IS R().
                                       SET BIT15=0
0036  03312 035730 FIX     STB  TEMP   SAVE ADDRESS
0037  03313 045727         ISZ  T
0038  03314 067331         JMP  DOIT   T ≠ −1 CARD IS IN PROCESS AND RECORDING
                                       CAN CONTINUE
0039  03315 020356         LDA  DATST  T=−1
0040  03316 063500         JSM  CRDIN  CHECK CARD IN
0041  03317 076150         RZB  JMP1   B=0,STOP KEY ENTERED. ABORT OPERATION
0042  03320 020401 JMP20   LDA  DP3      ERROR ≠3 SENT TO CALLING ROUTINE
                                       (A-REG)
0043  03321 170402         RET
0044*
0045  03322 070252 JMP1    SAM  NTYET  B≠0,A—NOT PROTECT,JUMP TO DELAY
0046  03323 020405         LDA  SETUP    CARD IS PROTECT
0047  03324 063553         JSM  OUT2   START SHUT DOWN PROCESS
0048  03325 020402 JMP2    LDA  DP2      ERROR ≠ 2
0049  03326 170402         RET
0050*
0051  03327 020513 NTYET   LDA  WAIT   DELAY CARD.1 INCH.PREVENTS ERRORS WHEN
0052  03330 072030         RIA  *        LOADING DUE TO SLIGHT POSITIONAL
                                         DIFFERENCES OF THE CARD-IN SENSO
0053  03331 021731 DOIT    LDA  WDCNT  1ST WORD ON CARD IS WORD COUNT. FOUR
0054  03332 063523         JSM  WRDST    TYPES OF BLOCKS POSSIBLE: CONFIGURA-
                                         TION,PROGRAM,ABCXYZ,R
0055  03333 021731         LDA  WDCNT  CHECK FOR TYPE OF BLOCK.
0056  03334 070513         SAP  JMP3   WORD COUNT POSITIVE, PROGRAM BLOCK
```

```
0057  03335 021733          LDA ADMOD  COUNT NEGATIVE THUS BLOCK IS DATA
0058  03336 063523          JSM WRDST  STORE ADDRESS OF BLOCK.
0059  03337 063360          JSM XDUMP  RECORD BLOCK OF INFORMATION
0060  03340 025733          LDB ADMOD  GET ORIGINAL ADDRESS
0061  03341 074213          SBP LEAVE    IF + TEMP OKAY
0062  03342 025730          LDB TEMP     IF -,SET TEMP BIT 15=1. NECESSARY IF
0063  03343 076052          SBM RO,S     ADDITIONAL CARDS ARE NEEDED FOR COM-
0064  03344 035730 RO       STB TEMP     PLETE RECORDING OF THE INFORMATION
                                         BLOCK.
0065  03345 170402 LEAVE    RET
0066*
0067*******************************
0068*
0069  03346 070076 JMP3     TCA        BLOCK IS PROGRAM.MAKE COUNT NEGATIVE
0070  03347 031731          STA WDCNT  STORE WORD COUNT
0071  03350 063360          JSM XDUMP  RECORD BLOCK OF INFORMATION
0072  03351 025730          LDB TEMP   GET ADDRESS OF NEXT PROGRAM WORD
                                           LOCATION
0073  03352 035546          STB CMA    SET THIS ADDRESS IN CMA (SYSTEM PROGRAM
                                           COUNT)
0074  03353 070210          SZA *+4    IF A≠0 MORE CARDS ARE NECESSARY
0075  03354 025731 JMP17    LDB WDCNT  MORE CARD NECESSARY
0076  03355 074076          TCB        SET WORD COUNT POSITIVE
0077  03356 035731          STB WDCNT  SAVE FOR NEXT CARD
0078  03357 170402          RET
0079*
0080*******************************
0081*
0082  03360 121730 XDUMP    LDA TEMP,I UTILITY ROUTINE FOR RECORDING BLOCK
                                           OF INFORMATION
0083  03361 063523          JSM WRDST  CALL ROUTINE TO PLACE A WORD ON THE
                                           CARD
0084  03362 045730          ISZ TEMP   SET ADDRESS FOR NEXT WORD
0085  03363 045731          ISZ WDCNT  CHECK WORD COUNT
0086  03364 067370          JMP XXDMP  MORE WORDS
0087  03365 070742 OUT      SAR 16     ALL WORDS RECORDED,SET A=0
0088  03366 074742          SBR 16
0089  03367 170402          RET
0090*
0091  03370 063516 XXDMP    JSM GET    GET SENSOR DATA
0092  03371 070146          RAR 4      POSITION END-OF CARD-BIT (4) IN BIT15
0093  03372 071312          SAM XDUMP  IF BIT15=1,NO END OF CARD: RECORD NEXT
                                           WORD
0094  03373 020403 E1       LDA DP1    ERROR ≠1
0095  03374 170402          RET
0096*
0097*******************************
0098*
0099  03375 063516 .END.    JSM GET    GET SENSOR INFORMATION
0100  03376 070344          SAL 9        POSITION END-OF-CARD FOR LOADING
                                           (BIT6) INTO BIT15
0101  03377 071712          SAM .END.  IF=1 CARD IS BLOCKING LIGHT RECYCLE
0102  03400 020000          LDA DELAY  CARD BEYOND SENSOR
0103  03401 072030          RIA *        DELAY
0104  03402 020361          LDA OFF    LOAD MOTOR OFF BIT
0105  03403 063553          JSM OUT2   OUTPUT TO INTERFACE
0106  03404 173741          CLF 1      TURN-ON INTERRUPT
0107  03405 170402 FINSH    RET
0108*
0109*******************************
0110*
0111  03406 070742 .LOD.    SAR 16     ENTRY FOR READING A MAG CARD
0112  03407 031727          STA T        SET T=0
```

```
0113  03410 020405          LDA SETUP     A-REG HAS CONDITION FOR READING
                                             (MOTOR-ON& INHIBIT)
0114  03411 063500          JSM CRDIN
0115  03412 075050          SZB E1        IF B=0 STOP KEY
0116*
0117  03413 063556 YESCD    JSM DATIN     GET FIRST WORD FROM CARD
0118  03414 076210          RZB JMP11     IF B#0;A-REG HAS WORD COUNT
0119  03415 021731          LDA WDCNT     GET WORD COUNT
0120  03416 071350          SZA FINSH     IF ZERO RETURN TO CALLING ROUTINE
0121  03417 067325          JMP JMP2      MORE WORDS BUT EOC;ERROR #2
0122*
0123  03420 070352 JMP11    SAM JMP8      TEST WORD COUNT TO DETERMINE TYPE
                                             OF BLOCK;IF +
0124  03421 070076          TCA           PROGRAM;SET WORD COUNT NEGATIVE
0125  03422 031731          STA WDCNT
0126  03423 020405          LDA DM1
0127  03424 031727          STA T         SET T=-1 TO DENOTE PROGRAM BEING
                                             LOADED
0128  03425 021546          LDA CMA       GET ADDRESS OF WHERE TO START LOAD
                                             PROGRAM
0129  03426 067442          JMP MAXAD     CHECK IF PROGRAM WILL FIT IN MEMORY
0130*
0131*******************************
0132*
0133  03427 031731 JMP8     STA WDCNT     THE CARD TO BE READ HAS DATA
0134  03430 063556          JSM DATIN     NEXT WORD ON CARD IS ADDRESS,GET IT
0135  03431 070473          SAP MAXAD,C   IF BIT15=0 DATA IS A,B,C,X,Y,Z CONTINUE
                                             WITH READING
0136  03432 021731          LDA WDCNT     BIT15=1,ADDRESS NORMALIZED TO SIZE
                                             OF MACHINE
0137  03433 001377          ADA MAW       ADD -COUNT TO MAXIMUM ADDRESS
                                             WORD (MAW)
0138  03434 031730          STA TEMP      SET NEW ADDRESS FOR DATA IN TEMP
0139  03435 070076          TCA           CHECK TO SEE IF ADDRESS IS IN SYSTEM
                                             INTER STACK
0140  03436 001542          ADA ENDS
0141  03437 001615          ADA BSTK      (ENDS+BSTK)-TEMP
0142  03440 070413          SAP OVFLO     IF RESULT POSITIVE ERROR, JUMP TO
                                             OVERFLOW
0143  03441 067451          JMP LOAD      READ BLOCK
0144*
0145  03442 031730 MAXAD    STA TEMP      CHECK FOR PROGRAM FIT IN MEMORY.
                                             SAVE LOAD ADDRESS
0146  03443 001615          ADA BSTK      WORDS NECESSARY FOR INTERPRETER
                                             STACK
0147  03444 070056          CMA
0148  03445 001377          ADA MAW
0149  03446 001731          ADA WDCNT     MAW>=LOAD ADDRESS+#WORDS+BSTK
0150  03447 070113          SAP LOAD      IF TRUE JUMP TO READ CARD
0151  03450 067320 OVFLO    JMP JMP20       ELSE SET ERROR # 3
0152*
0153*******************************
0154*
0155  03451 063556 LOAD     JSM DATIN     UTILITY ROUTINE WHICH READS A BLOCK
                                             OF INFORMATION
0156  03452 076250          RZB JMP9      GET A WORD. IF B=0,END-OF-CARD
                                             FOUND
0157  03453 045727          ISZ T
0158  03454 067325          JMP JMP2      T#-1,RETURN ERROR #2,ALSO MUST BE
                                             DATA DON'T FIX ADD.
0159  03455 063472          JSM FIXAD     T=-1,READING PROGRAM FIX ADDRESS
                                             (CMA AND ENDS)
0160  03456 067325          JMP JMP2
0161*
0162  03457 025730 JMP9     LDB TEMP
0163  03460 017603          CPB ASE       IS STORE ADDRESS SECURE WORD?
```

```
0164  03461  067463          JMP  *+2      YES SKIP STORE
0165  03462  131730          STA  TEMP,I   CONTINUE TO STORE WORDS
0166  03463  045730          ISZ  TEMP     ADJUST POINTER TO NEXT MEMORY
                                            LOCATION
0167  03464  045731          ISZ  WDCNT    CHECK WORD COUNT
0168  03465  067451          JMP  LOAD       NOT ZERO. CONTINUE LOADING
0169  03466  045727          ISZ  T        ZERO,CHECK T=-1 PROGRAM, T#-1 DATA
0170  03467  067413          JMP  YESCD    DATA;BLOCK COMPLETE
0171  03470  063472          JSM  FIXAD    PROGRAM BLOCK COMPLETED; FIX CMA
                                            AND ENDS
0172  03471  067413          JMP  YESCD    RETURN
0173*
0174  03472  021730 FIXAD    LDA  TEMP     GET NEXT STOREAGE ADDRESS
0175  03473  031546          STA  CMA      PLACE IN SYSTEM PROGRAM POINTER
0176  03474  031542          STA  ENDS       SET ENDS (POINTER TO END-OF-PROGRAM
                                            AREA)
0177  03475  020405          LDA  DM1
0178  03476  131542          STA  ENDS,I   STORE -1 AT END OF PROGRAM
0179  03477  170402          RET           RETURN TO CALLING ROUTINE
0180*
0181*******************************
0182*
0183  03500  172741 CRDIN    STF  1        CHECK FOR CARD-IN.  TURN-OFF
                                            INTERRUPT
0184  03501  063533          JSM  OUT2     SET CONDITION IN A-REG INTO MAG CARD
                                            INTERFACE
0185  03502  063516 LOOK     JSM  GET      GET SENSOR INFORMATION
0186  03503  070146          RAR  4        PLACE CARD-IN-BIT (4) IN LSB
0187  03504  070211          SLA  JMP10    BIT0=0 CARD NOT IN JUMP TO CHECK IF
                                            STOP
0188  03505  070046          RAR  2          CHECK IF LEADING EDGE OF CARD
0189  03506  070111          SLA  JMP10
0190  03507  170402          RET             CARD IS IN!
0191  03510  025701 JMP10    LDB  .WKC     LOAD KEYCODE
0192  03511  014537          CPB  STPKY    IS IT STOP?
0193  03512  074742          SBR  16       YES, B-REG=0
0194  03513  077350          RZB  LOOK     B#0, RECYCLE (LOOK FOR CARD IN)
0195  03514  020405          LDA  SETUP    B=0 SET UP CONDITION TO GET CARD
                                            THRU CARD READER
0196  03515  067553          JMP  OUT2     INHIBIT AND TURN ON MOTOR
0197*
0198*******************************
0199*
0200  03516  020423 GET      LDA  MASK       ALL 1's IN MOST SIGNIFICANT RITS OF I/O
                                            REGISTER ALLO
0201  03517  172141          OTA  01         LOADING OF THE SENSOR'S AND DATA
                                            INFORMATION
0202  03520  172741          STF  01         FROM THE MAG CARD INTERFACE
0203  03521  172241          LIA  01       MOVE 12 BIT WORD FROM I/O REGISTER
                                            TO A-REG
0204  03522  170402          RET
0205*
0206*******************************
0207*
0208  03523  024412 WRDST    LDB  M6       UTILITY ROUTINE USED TO STORE 16 BITS
                                            ON A MAG CARD
0209  03524  035734          STB  BTCNT      SET BYTE COUNT TO 6 (3BITS/BYTE)
0210  03525  024522          LDB  FMTOT    LOAD B-REG ALTERNATING BIT PATTERN
0211  03526  070056          CMA           BECAUSE OF INPUT REQUIREMENTS
0212  03527  031732 BTOUT    STA  HOLD       (READING) COMPLEMENT DATA. SAVE IT
0213  03530  050375          AND  DP7      GET 3 BITS
0214  03531  040423          IOR  MASK     SET MSB FOR ADDRESS OF MAG CARD
0215  03532  063550          JSM  OUT1     OUTPUT WORD TO INTERFACE
0216  03533  020422          LDA  422B     -25
0217  03534  072030          RIA  *        DELAY FOR STROBE WHICH PLACE IT .003
                                            INCH FARTHER A
```

```
0218  03535  176141          OTB  01     SET STROBE BIT INTO BIT3 (ALTERNATING
                                           BIT PATTERN)
0219  03536  172742          STF  02     CLOCK FOR STROBE BIT LATCH
0220  03537  074006          RBR  1      POSITION FOR NEXT STROBE BIT
0221  03540  020422          LDA  422B   -25
0222  03541  072030          RIA  *      DELAY FOR NEXT DATA ENTRY (.003 INCH)
0223  03542  045734          ISZ  BTCNT  CHECK BYTE COUNTER
0224  03543  067545          JMP  JMP14  NOT ZERO, GO POSITION FOR NEXT BYTE
0225  03544  170402          RET         ZERO, RETURN WORD RECORDED
0226  03545  021732  JMP 14  LDA  HOLD
0227  03546  070106          RAR  3      POSITION TO OUTPUT NEXT BYTE
0228  03547  067527          JMP  BTOUT  RECYCLE
0229*
0230  03550  172141  OUT1    OTA  01     OUTPUT DATA TO I/O REGISTER
0231  03551  172602          STC  02     LATCH DATA INTO INTERFACE CARD
0232  03552  170402          RET
0233*
0234*********************************
0235  03553  063550  OUT2    JSM  OUT1   CALL OUTPUT TO I/O REGISTER ROUTINE
0236  03554  172742          STF  02     SET STROBE LATCH
0237  03555  170402          RET
0238*
0239*********************************
0240*
0241  03556  070742  DATIN   SAR  16     UTILITY ROUTINE FOR GET A 16 BIT WORD
                                           FROM CARD
0242  03557  031732          STA  HOLD   CLEAR BUFFER WORD
0243  03560  024412          LDB  M6
0244  03561  035734          STB  BTCNT  SET BYTE COUNTER TO SIX
0245  03562  173742  WDBLD   CLF  02     CLEAR FLAG FLIP-FLOP ON CARD
                                           INTERFACE
0246  03563  063516  FLAG    JSM  GET    GET DATA WORD
0247  03564  172502          SFS  02     IS DATA PRESENT?
0248  03565  067600          JMP  EOCL   NO, GO CHECK FOR END-OF-CARD
0249  03566  063516          JSM  GET
0250  03567  070144          SAL  13     YES,PLACE DATA BITS (3) IN MSD
0251  03570  041732          IOR  HOLD   OR IN DATA BITS FROM HOLD
0252  03571  045734          ISZ  BTCNT  CHECK BYTE COUNT
0253  03572  067575          JMP  JMP4   NOT ZERO, MORE BYTES NECESSARY
0254  03573  070646          RAR  14     ZERO,NECESSARY TO OBTAIN CORRECT
                                           DATA FORMAT
0255  03574  170402          RET
0256*
0257  03575  070106  JMP4    RAR  3      READY HOLD TO RECEIVED NEXT 3 BIT
                                           BYTE
0258  03576  031732          STA  HOLD   SAVE IT
0259  03577  067562          JMP  WDBLD  JUMP TO GET NEXT BYTE
0260*
0261  03600  070306  EOCL    RAR  7      POSITION END-OF-CARD BIT INTO BIT15
0262  03601  071112          SAM  FLAG   IF BIT15=1 MORE CARD LEFT, CHECK FOR
                                           DATA STROBE
0263  03602  067366          JMP  OUT+1  END-OF-CARD SET B=0
0264*
0265  03603  001625  ASE     DEF  1625B
0266*
0267  03604  000000          BSS  1      CHECKSUM
0268  03605  003306          DEF  .REC.  LINKAGE TO MAG CARD DRIVERS
0269  03606  003375          DEF  .END.  *
0270  03607  003406          DEF  .LOD.  *
0271*
0272*********************************
0273*
0274  00000          DELAY   EQU  0
0275  01546          CMA     EQU  1546B
0276  01730          TEMP    EQU  1730B
```

```
0277   01731          WDCNT   EQU  1731B
0278   01732          HOLD    EQU  1732B
0279   01733          ADMOD   EQU  1733B
0280   00513          WAIT    EQU  513B
0281   01734          BTCNT   EQU  1734B
0282   01727          T       EQU  1727B
0283   00356          DATST   EQU  356B
0284   01615          BSTK    EQU  1615B   # OF WORDS REGUIRED FOR INTERPRETER
0285*    57
0286   00405          SETUP   EQU  405B
0287*    77
0288   01377          MAW     EQU  1377B
0289   00537          STPKY   EQU  537B
0290*    22
0291   00522          FMTOT   EQU  522B
0292*    052525
0293   00423          MASK    EQU  423B
0294*    177740
0295   00412          M6      EQU  412B
0296*    177712
0297   00361          OFF     EQU  361B
0298   00403          DP1     EQU  403B
0299   00402          DP2     EQU  402B
0300   00401          DP3     EQU  401B
0301   00400          DP4     EQU  400B
0302   00376          DP6     EQU  376B
0303   00375          DP7     EQU  375B
0304   01701          .WKC    EQU  1701B
0305   00405          DM1     EQU  405B
0306   01542          ENDS    EQU  1542B
0307*
0308   00303                  ORG  303B
0309   00303  003365          DEF  OUT
0310   00412                  ORG  412B
0311   00412  177772          DEC  -6
0312   00421                  ORG  421B
0313   00421  177715          OCT  177715
0314   00522                  ORG  522B
0315   00522  052525          OCT  052525
0316                          END
*    NO ERRORS*
```

```
0001                        ASMB,A,L            COMPLIER SUBROUTINES
0002  01266                       ORG  1266B
0003*
0004* RECOVERABLE ERROR (USED BY MATH ROUTINES ONLY)
0005*
0006  01266  021430  RECRR  LDA  FLAG  SET FLAG 15
0007  01267  072053        SAP  *+1,S  FOR MATH ERRORS
0008  01270  031430        STA  FLAG  SO USER CAN TEST LATER
0009  01271  070744        SAL  1  CHECK FLAG 14
0010  01272  070252        SAM  *+5  AND RETURN IF SET
0011  01273  065351        JMP  BAD  ELSE SET ERROR
0012*
0013* DISALLOW IMPLIED STORAGE
0014*
0015  01274  025601  DISIS  LDB  STAT  (SEE PAGE 5 FOR DETAILS)
0016  01275  074071        SLB  *+1,C
0017  01276  035601        STB  STAT
0018  01277  170402        RET
0019*
0020* BUILD COMPILED STRING
0021*
0022  01300  014452  CL    CPB  CNFQ  ENTER HERE FOR LITERAL STRING
0023  01301  045602        ISZ  G  INCREMENT G IF END OF QUOTE
0024  01302  014457        CPB  CNFY  ERROR IF EOL, SINCE A QUOTE
0025  01303  065317        JMP  E1  CANNOT BE CARRIED INTO NEXT LINE
0026*
0027  01304  021544  CP1   LDA  SKEY  USUALLY ENTERS HERE
0028  01305  025545  CP2   LDB  PSP  PSP POINTS TO NEXT AVAILABLE WORD
0029  01306  074233        SBP  *+4,C  SIGN TELLS WHICH HALF TO USE
0030  01307  074617        IOR  B,I
0031  01310  074557        STA  B,I  IF SET, USE RIGHT HALF
0032  01311  076230        RIB  *+4
0033  01312  070444        SAL  7
0034  01313  074557        STA  B,I  IF CLEAR, USE LEFT HALF
0035  01314  076053        SBP  *+1,S
0036  01315  035545        STB  PSP  STORE UPDATED POINTER
0037  01316  065362        JMP  EXIT  AND RETURN THRU NORMAL EXIT
0038*
0039  01317  024403  E1    LDB  DP1  SYNTAX ERROR
0040  01320  065351        JMP  BAD
0041*
0042*       THESE BASE-PAGE ROUTINES ARE AVAILABLE TO THE REST OF
0043*       THE SYSTEM FOR UTILITY PURPOSES
```

```
0045*
0046*  CONVERT-TO-INTEGER
0047*
0048   01321  121556  FIXPT   LDA  AP1,I    AP1 POINTS TO NUMBER TO BE CONVERTED
0049   01322  070073          SAP  *+1,C    CLEAR TEMPORARY BIT
0050   01323  024450          LDB  ADR2     ADDRESS OF AR2
0051   01324  170004          XFR           MOVE THE NUMBER TO AR2
0052   01325  074742          SBR  16       INITIAL FIXED-POINT RESULT=0
0053   01326  120450          LDA  ADR2,I   LOOK AT EXPONENT
0054   01327  070712          SAM  *+14     DONE IF 10'S EXPONENT IS <0
0055   01330  000427          ADA  DM256    ELSE DECR 10'S EXPONENT
0056   01331  031721          STA  H1       AND SAVE IT
0057   01332  074117          LDA  B
0058   01333  000363          ADA  M3276    TEST FOR INTEGER OVERFLOW
0059   01334  070613          SAP  *+12     SKIP IF OVERFLOW
0060   01335  074744          SBL  1        MULTIPLY PREVIOUS RESULT BY 10
0061   01336  035717          STB  T1       2 * RESULT
0062   01337  074704          SBL  2        8 * RESULT
0063   01340  005717          ADB  T1
0064   01341  171400          LS            SHIFT AR2 LEFT
0065   01342  070037          ADB  A        ADD IN NEXT DIGIT
0066   01343  021721          LDA  H1       CURRENT ANSWER IS IN B-REGISTER
0067   01344  071213          SAP  *-12
0068   01345  070111          SLA  *+2      TEST SIGN OF MANTISSA
0069   01346  074076          TCB           REPRESENT NEGATIVE NUMBER AS 2'S
                                            COMPLEMENT
0070   01347  170402          RET           RETURN WITH RESULT IN B-REGISTER
0071*
0072   01350  024372          LDB  DP10     ERROR
0073*
0074*      ERROR #10 (INTEGER OVERFLOW) WILL USUALLY BE MASKED
0075*      BY SOME OTHER ERROR, SUCH AS ILLEGAL SUBSCRIPT
0076*
0077*      THIS MAY NOT BE THE CASE IF AN OPTION BLOCK USES THE
0078*      CONVERT-TO-INTEGER ROUTINE
0079*
0080*  FLOATING-POINT NUMBER FORMAT:
0081*
0082*      WORD 1:    EEEEEEEE --------          S
0083*      WORD 2:    DDDD DDDD DDDD DDDD    MOST SIGNIFICANT DIGITS
0084*      WORD 3:    DDDD DDDD DDDD DDDD
0085*      WORD 4:    DDDD DDDD DDDD DDDD    HIGH-CORE END OF #
```

```
0087*
0088*  ERROR EXIT
0089*
0090   01351  045603  BAD    ISZ   ERRS   SET ERRORS FLAG
0091   01352  074644         SBL   3      POSITION THE ERROR BITS
0092   01353  021705         LDA   .WMOD  FOR INSERTION INTO .WMOD
0093   01354  050513         AND   ERM2   "AND" OUT OLD ERROR NUMBER
0094   01355  074217         IOR   B      "OR" IN NEW ERROR NUMBER
0095   01356  031705         STA   .WMOD  STORE THE INDICATOR WORD .WMOD
0096   01357  024405         LDB   DM1    INDICATE ERROR
0097   01360  170402         RET          TO CALLING PROGRAM
0098*
0099   01361  000000         NOP          RESERVED FOR CHECKSUM WORD
0100*
0101*  NORMAL EXIT
0102*
0103   01362  021603  EXIT   LDA   ERRS   ARE THERE ANY PREVIOUS ERRORS?
0104   01363  073610         RZA   *-4    YES
0105   01364  025566         LDB   CP     NO
0106   01365  074073         SBP   *+1,C  COMPARE CP AND PSP FOR A
0107   01366  021545         LDA   PSP    POSSIBLE OVERLAP OF THE SAME
0108   01367  070073         SAP   *+1,C  R/W MEMORY AREA
0109   01370  070056         CMA          BY ARITHMETICALLY COMPARING
0110   01371  074017         ADA   B      THE TWO ADDRESSES
0111   01372  070152         SAM   *+3    THEY ARE ABOUT TO OVERLAP
0112   01373  074742         SBR   16     ALL OK
0113   01374  170402         RET
0114*
0115   01375  024373  E9A    LDB   DP9    ONE OF TWO CIRCUMSTANCES UNDER
0116   01376  065351         JMP   BAD    WHICH AN OVERLAP OCCURS
0117*
0118*  EXIT FROM THE COMPILER IS ALWAYS THROUGH ONE OF THE ABOVE
0119*  TWO ROUTINES
0120*
0121*  THE NORMAL EXIT MUST CHECK FOR PREVIOUS ERRORS
0122*  RATHER THAN ASSUME THAT ALL IS O.K. SO FAR
0123*
0124*  THIS IS MAINLY BECAUSE AN ERROR MAY HAVE OCCURED IN A
0125*  SUBROUTINE INTERNAL TO THE COMPILER, IN WHICH CASE THE
0126*  ERROR RETURN IS TO THE COMPILER
```

```
0128*  COMPILER BEGINS HERE
0129*
0130*  CONVERTS INFIX NOTATION TO POLISH NOTATION
0131*
0132*  FLAGS:
0133*
0134*      G       IF ZERO, PROCESS INPUT CHARACTER NORMALLY
0135*              IF NONZERO, GO TO SPECIAL ROUTINE INDICATED
0136*              IN THE TABLE TTB2 (SEE PAGE 40)
0137*
0138*  PS          STATE VARIABLE FOR SYNTAX CHECKING
0139*              GOVERNED BY THE SYNTAX TABLE ON THE NEXT PAGE
0140*
0141*  STAT        CONTROLS USE OF COMMA, QUOTE, IMPLIED STORAGE
0142*              BIT 0 SET ALLOWS IMPLIED STORAGE
0143*              BIT 1 SET ALLOWS COMMA
0144*              BIT 2 SET ALLOWS QUOTE
0145*
0146*  ERRS        SIGNALS AT LEAST ONE ERROR HAS BEEN DETECTED
0147*
0148*  DP          SEVERAL FLAGS USED AT VARIOUS
0149*  SD          STAGES OF NUMBER
0150*  ES          SYNTAX CHECKING
0151*
0152*  THE CONVERSION ALGORITHM:
0153*
0154*      ALL OPERANDS ARE MOVED DIRECTLY TO THE POLISH OUTPUT
0155*      STRING VIA THE ROUTINE CP1 (SEE PAGE 2)
0156*
0157*      CP2 IS A SPECIAL ENTRY USED WHEN A CHARACTER OTHER THAN
0158*      THE CONTENTS OF SKEY NEEDS TO BE MOVED OUT
0159*
0160*      EACH OPERATOR IS ASSIGNED A COMPARE PRIORITY AND A STACK
0161*      PRIORITY
0162*
0163*      THE COMPARE PRIORITY OF THE NEWLY ENCOUNTERED OPERATOR
0164*      IS CHECKED AGAINST THE STACK PRIORITY OF THE OPERATOR
0165*      CURRENTLY ON TOP OF A LAST-IN FIRST-OUT STACK
0166*
0167*      IF THE COMPARE PRIORITY OF THE NEW OPERATOR IS GREATER
0168*      THAN THIS STACK PRIORITY, THE NEW OPERATOR IS ADDED TO
0169*      THE STACK
0170*
0171*      OTHERWISE THE OPERATOR ON TOP OF THE STACK IS REMOVED
0172*      FROM THE STACK AND PLACED IN THE POLISH OUTPUT STRING
0173*      AND THE RELATIVE PRIORITY TEST IS REPEATED
0174*
0175*      IN THE CASE OF GTO OR GSB AN EXTRA (BLANK) MACHINE WORD
0176*      IS PLACED IN THE POLISH STRING FOR USE AT EXECUTION TIME
0177*      WHEN A HIGH-SPEED ACCESS ADDRESS IS FILLED IN BY THE
0178*      INTERPRETER
```

```
0180*  SYNTAX TABLE:
0181*
0182*       THE PREVIOUSLY ENCOUNTERED CHARACTER (IF LEGAL) SETS THE
0183*       STATE VARIABLE PS TO 0, 1, 2, OR 4

            PS IS INITIALLY SET TO 3 (EXCEPT FOR ENTER MODE PS=5)

0184*       THE ROW CORRESPONDING TO PS IS CHECKED TO DETERMINE WHETHER
0185*       THE NEWLY ENCOUNTERED CHARACTER IS LEGAL
0186*
0187*       IF SO, PS IS RESET ACCORDINGLY
0188*       OTHERWISE THE CHARACTER IS REJECTED AS A SYNTAX ERROR
0189*

0190*                    OPND  (    FCN  )    SBO  +    -    ,    ;    QUOT
0191*  OLD    NEXT
0192*  CHAR   STATE
0193*
0194*  OPND   0      IM    IM   IM   OK   OK   OK   OK   CHK  OK   ERR
0195*  )
0196*
0197*  (      1      OK    OK   OK   ERR  ERR  U+   U-   ERR  ERR  CHK
0198*  ,
0199*
0200*  QPRTR  2      OK    OK   OK   ERR  ERR  U+   U-   ERR  ERR  ERR
0201*
0202*  INITL  3      OK    OK   OK   ERR  ERR  U+   U-   ERR  DEL  OK
0203*  STATE
0204*
0205*  PROC   4      OK    OK   OK   ERR  ERR  U+   U-   ERR  OK   CHK
0206*
0207*  INITL  5      (INITIAL STATE FOR ENTER STMTS -- SOME PROC IN ERROR)
0208*  STATE
0209*  ENTER
0210*
0211*  FCN    6      OK    OK   OK   ERR  ERR  ERR  ERR  ERR  ERR  ERR
0212*  ->
0213*
0214*       IM  = IMPLIED MULTIPLY
0215*
0216*       OK  = ALL OK SO FAR
0217*
0218*       CHK = CHECK STAT
0219*
0220*       ERR = ERROR
0221*
0222*       DEL = DELETE SUPERFLUOUS U+
0223*
0224*       U-  = CHANGE TO U-
0225*
0226*       U+  = DEL
0227*
0228*       A TABLE OF STACK AND COMPARE PRIORITIES IS PRESENTED ON
0229*       PAGE 14
```

```
0231   04055                    ORG 4055B
0232*
0233*  COMPILER SETUP
0234*
0235   04055  024535 CMSET      LDB TOPS      THIS SECTION INITIALIZES CERTAIN
0236   04056  035545            STB PSP       FLAGS AND POINTERS
0237   04057  024456            LDB OPST
0238   04060  035541            STB OP
0239   04061  074742            SBR 16
0240   04062  035603            STB ERRS
0241   04063  135541            STB OP,I      THEN INITIALIZES THE REMAINDER
0242   04064  066273            JMP CR
0243*
0244*  IMPLIED MULTIPLY
0245*
0246   04065  031606 CM          STA PS        AN IMPLIED MULTIPLY MUST BE
0247   04066  020017            LDA IMUL      INSERTED IN THE STRING
0248   04067  024372            LDB AMUL
0249   04070  062317            JSM CCI
0250*
0251*  COMPILER ENTRY
0252*
0253   04071  025540 CMPLR       LDB CN        FIRST LOOK AT THE CONTROL #
0254   04072  014355            CPB DP48      IGNORE CONTROL # 48
0255   04073  065373            JMP EXIT+9
0256   04074  014351            CPB DP63      IGNORE CONTROL # 63
0257   04075  065373            JMP EXIT+9
0258   04076  014353            CPB DP52      ERROR IF CONTROL # 52
0259   04077  065317            JMP E1
0260*
0261   04100  021602            LDA G         THEN LOOK AT G
0262   04101  072210            RZA *+4
0263*
0264   04102  021606            LDA PS        IF G IS ZERO THEN BRANCH TO A
0265   04103  007720            ADB TAB1      ROUTINE DETERMINED BY CN
0266   04104  074737            JMP B,I       VIA TABLE TAB1
0267*
0268   04105  003721            ADA TAB2      OTHERWISE BRANCH VIA TABLE TAB2
0269   04106  070737            JMP A,I       TO A SPECIAL ROUTINE
0270*
0271*  GAZINTA
0272*
0273   04107  061274 GAZ         JSM DISIS     NO IMPLIED STORAGE IF GAZINTA
0274   04110  073350 UPARO       RZA *-9
0275   04111  066305            JMP CPS6
0276*
0277*  STRICTLY BINARY OPERATOR
0278*
0279   04112  073250 SBO         RZA *-11      THIS INCLUDES BINARY OPERATORS
0280   04113  066311            JMP CPS2      EXCEPT FOR + AND -
```

```
0282*
0283*  PLUS
0284*
0285   04114  071710  PLU     SZA  SBO     IT MAY BE A U+. A=0 THEN BINARY
                                            OPERATOR
0286*
0287   04115  164303  IGNR    JMP  OUT,I   IGNORE ROUTINE,SET A&B-REGS ZER0
0288*
0289*  MINUS
0290*
0291   04116  071610  MIN     SZA  SBO     IT MAY BE A U-, A=0 THEN BINARY
                                            OPERATOR
0292*
0293   04117  010376          CPA  DP6     U- CANNOT FOLLOW
0294   04120  065317          JMP  E1
0295*
0296   04121  020402          LDA  DP2     CHANGE THE REGULAR - TO A U-
0297   04122  031606          STA  PS      BY SWAPPING CODES
0298   04123  020155          LDA  IUMN
0299   04124  024506          LDB  AUMN
0300   04125  066317          JMP  CC1
0301*
0302*  FUNCTION, SUBSCRIPTED VARIABLE
0303*
0304   04126  070310  FCN     SZA  *+6     THE R-REGISTERS ARE TREATED THE
0305   04127  066305          JMP  CPS6    SAME AS THE FUNCTIONS
0306*
0307*  OPERAND
0308*
0309   04130  070210  OPD     SZA  *+4     OPERANDS GO DIRECTLY TO THE
0310   04131  070742          SAR  16      POLISH OUTPUT STRING
0311   04132  031606          STA  PS      (SEE PAGE 5)
0312   04133  065304          JMP  CP1
0313*
0314   04134  020402          LDA  DP2     INSERT AN IMPLIED MULTIPLY
0315   04135  066065          JMP  CM      ACCORDING TO SYNTAX TABLE
```

```
0317*
0318* (
0319*
0320   04136  070650 LBR    SZA  *+13      AGAIN THERE IS THE POSSIBILITY
0321   04137  020403        LDA  DP1       OF IMPLIED MULITPLY
0322   04140  031606        STA  PS
0323*
0324   04141  125541        LDB  OP,I      WHAT IS INSIDE THE BRACKETS?
0325   04142  074604        SBL  4
0326   04143  074113        SBP  *+2
0327   04144  020402        LDA  DP2
0328   04145  025601        LDB  STAT
0329   04146  074404        SBL  8
0330   04147  004503        ADB  ALPR
0331   04150  031601        STA  STAT
0332   04151  021561        LDA  CODE
0333   04152  066317        JMP  CC1
0334*
0335   04153  020403        LDA  DP1       INSERT AN IMPLIED MULTIPLY
0336   04154  066065        JMP  CM        ACCORDING TO SYNTAX TABLE
0337*
0338* )
0339*
0340   04155  072410 RBR    RZA  *+8       THE RIGHT BRACKET MUST
0341   04156  066315        JMP  CC         FOLLOW AN OPERAND
0342*
0343* ,
0344*
0345   04157  010403 COM    CPA  DP1
0346   04160  065317        JMP  E1        ERROR IF PS=1
0347   04161  021601        LDA  STAT      ALSO HAVE TO CHECK STAT
0348   04162  070002        SAR  1         BIT 1 MUST BE SET
0349   04163  070111        SLA  *+2
0350   04164  066313        JMP  CPS1      THE COMMA IS OK
0351*
0352   04165  065317        JMP  E1        THE COMMA IS ILLEGAL
```

```
0354*
0355*  FLAGS
0356*
0357   04166  024372  STF     LDB  DP10     SET G=10 SO NEXT KEYCODE CAN BE
                                             LOOKED AT
0358   04167  035602          STB  G
0359   04170  062420          JSM  CHKB1    CHECK FOR PROPER SYNTAX
0360   04171  066313          JMP  CPS1     KEY OKAY
0361*
0362   04172  014546  GE10    CPB  CNFG       IF SAME KEYCODE AS LAST TIME
                                               CHANGE CODE TO CLF
0363   04173  066175          JMP  *+2
0364   04174  066236          JMP  STZG       ELSE PROCESS PRESENT KEY
0365*
0366   04175  055777          DSZ  1777B    SETUP RETURN TO BUILD DISPLAY
0367   04176  163764          JSM  DRECP,I  MOVE POINTER OVER SFG
0368   04177  024371          LDB  ACLF
0369   04200  160354          JSM  354B,I   STORE CLF IN INPUT BUFFER
0370   04201  045541          ISZ  OP         REMOVE STF FROM STACK
0371   04202  070742          SAR  16
0372   04203  031602          STA  G
0373   04204  020371          LDA  ACLF
0374   04205  160254          JSM  SKCD,I   SETUP SKEY,CODE,AND CN
0375   04206  066315          JMP  CC
0376*
0377*  IF, JUMP
0378*
0379   04207  062422  IF      JSM  CHKB     IF AND JMP MUST BE FOLLOWED BY
0380   04210  066313          JMP  CPS1     AN ALGEBRAIC EXPRESSION
0381*
0382*  RETURN,STOP
0383*
0384   04211  024376  LR      LDB  DP6      WHEREAS THESE CODES MAY BE
0385   04212  035601          STB  STAT       COMPLETE IN THEMSELVES
0386*
0387*  RETURN,STOP,END
0388*
0389   04213  062422  RET     JSM  CHKB     NO COMMAS APPEAR WITH THESE CODES
0390   04214  066307          JMP  CPS4     BUT THEY MAY APPEAR ALONE
0391*
0392*  PRINT,DISPLAY
0393*
0394   04215  024376  PD      LDB  DP6
0395   04216  035601          STB  STAT
0396*
0397*  SPACE,FIXED,FLOAT,NORM,TRACE,END
0398*
0399   04217  062420  SFFNT   JSM  CHKB1
0400   04220  066307          JMP  CPS4
```

```
0402*
0403*  ENTER
0404*
0405   04221  062422  ENT    JSM   CHKB   ENTER DOES A LITERAL TRANSFER OF
0406   04222  020403         LDA   DP1    CHARACTERS TO THE STRING AREA
0407   04223  031602         STA   G      UNTIL THE END OF THE STATEMENT
0408   04224  065304         JMP   CP1
0409*
0410   04225  014454  GE1    CPB   CNFE   LOOK OUT FOR EOS OR EOL
0411   04226  066232         JMP   *+4      WHICH SIGNALS THE END OF
0412   04227  014457         CPB   CNFY   THE STATEMENT
0413   04230  066232         JMP   *+2      ALLOW EOL
0414   04231  065304         JMP   CP1    KEEP GOING
0415*
0416   04232  020507  SNLL   LDA   NULL   A NULL CHARACTER AT THE END
0417   04233  061305         JSM   CP2    SIGNALS THE REVERSE COMPILER
0418   04234  020403         LDA   DP1    SIGNALS END OF ENTER STMT
0419   04235  061305         JSM   CP2
0420   04236  070742  STZG   SAR   16     TO DO A LITERAL TRANSLITERATION
0421   04237  031602         STA   G      BACK TO ALGEBRAIC FORM
0422   04240  066071         JMP   CMPLR
```

```
0424*
0425*  END-OF-LINE
0426*
0427   04241  062254  EOL     JSM  EOS      SAME AS EOS IN MANY WAYS
0428*
0429   04242  120535          LDA  TOPS,I   SET THE SIGN BIT AT THE
0430   04243  072053          SAP  *+1,S    START OF THE POLISH OUTPUT
0431   04244  130535          STA  TOPS,I
0432   04245  025545          LDB  PSP      MAKE SURE THAT PSP POINTS TO
0433   04246  074133          SBP  *+2,C    THE BEGINNING OF A LINE
0434   04247  074070          SIB  *+1
0435   04250  035545          STB  PSP
0436   04251  020405          LDA  DM1      BOUNDARY FOR COMPILER OUTPUT AREA
0437   04252  131545          STA  PSP,I    OF THE POLISH STRING
0438   04253  065362          JMP  EXIT
0439*
0440*  ;
0441*
0442   04254  010401  EOS     CPA  DP3      DOES EOS BEGIN THE LINE?
0443   04255  066272          JMP  EO       YES
0444*
0445   04256  010377          CPA  DP5      NECESSARY BECAUSE ENTER IS INITAL PS=5
0446   04257  066272          JMP  EO
0447*
0448   04260  070104          SAL  14       IS THE LINE COMPLETE AS
0449   04261  070110          SZA  *+2      IT STANDS?
0450   04262  065317          JMP  E1       NO, ERROR
0451*
0452   04263  062315          JSM  CC
0453   04264  021601          LDA  STAT     IS IMPLIED STORAGE ALLOWED?
0454   04265  070251          SLA  *+5      NO
0455   04266  020505          LDA  AZRG     INSERT IMPLIED STORAGE
0456   04267  061305          JSM  CP2      USING THE CP2 ENTRY
0457   04270  020464          LDA  AGNZ     FIRST A Z AND THEN A GAZINTA
0458   04271  061305          JSM  CP2
0459*
0460   04272  061304  EO      JSM  CP1      NOW STORE THE ENDING
0461*
0462   04273  020401  CR      LDA  DP3      RESET INTERNAL FLAGS
0463   04274  025563          LDB  TSW      AND THIS STATEMENT IS DONE
0464   04275  014377          CPB  DP5
0465   04276  020377          LDA  DP5
0466   04277  031606          STA  PS
0467   04300  020403          LDA  DP1
0468   04301  031601          STA  STAT     THIS SECTION IS ALSO USED BY
0469   04302  070742          SAR  16       THE COMPILER SETUP SEQUENCE
0470   04303  031602          STA  G
0471   04304  065362          JMP  EXIT     (SEE LINES 228—238)
0472*
0473*  REQUIREMENTS ON THE FORMAT OF THE POLISH STRING ARE
0474*  FURTHER DESCRIBED IN OTHER LISTINGS
0475*
0476*  THE HIGHLIGHTS ARE:
0477*
0478*  1)  THE LIMITS OF THE PROGRAM AREA ARE BOUNDED BY MACHINE
0479*      WORDS CONTAINING –1

0480*
0481*  2)  EACH LINE BEGINS WITH A MACHINE WORD WITH THE SIGN
0482*      BIT SET
0483*
0484*  3)  THE LAST STATEMENT IN THE MAIN PROGRAM IS END-EOL
```

```
0486*
0487*  STACK ROUTINE
0488*
0489  04305 024376 CPS6   LDB  DP6
0490  04306 066314        JMP  *+6
0491  04307 024400 CPS4   LDB  DP4    FIRST RESET PS PROPERLY
0492  04310 066314        JMP  *+4    WITH THESE INITIAL ENTRY POINTS
0493  04311 024402 CPS2   LDB  DP2
0494  04312 066314        JMP  *+2    SET IT TO 1, 2, OR 4
0495  04313 024403 CPS1   LDB  DP1
0496  04314 035606        STB  PS
0497*
0498  04315 021561 CC     LDA  CODE   GET THE INTERNAL AND EXTERNAL
0499  04316 025544        LDB  SKEY   FORMS OF THE CODE
0500*
0501  04317 031716 CC1    STA  THIS   AND TEMPORARILY SAVE THEM
0502  04320 035717        STB  T1     FOR LATER USE
0503*
0504  04321 050365 CCB    AND  DP15   COMPARE RELATIVE PRIORITIES
0505  04322 125541        LDB  OP,I   BY SUBTRACTING ONE FROM THE
0506  04323 074542        SBR  12     OTHER
0507  04324 074056        CMB
0508  04325 070037        ADB  A
0509  04326 121541        LDA  OP,I   ALSO REMEMBER WHAT IS ON
0510  04327 050345        AND  RH     TOP OF THE STACK RIGHT NOW
0511  04330 074453        SBP  *+9
0512*
0513  04331 010503        CPA  ALPR   UNSTACK A CHARACTER
0514  04332 066337        JMP  E3     SINCE NEW PRIORITY IS NOT
0515  04333 061305        JSM  CP2    HIGHER THAN TOP OF STACK
0516  04334 045541        ISZ  OP
0517  04335 021716        LDA  THIS
0518  04336 066321        JMP  CCB    LOOP BACK TO REPEAT THE TEST
0519*
0520  04337 024401 E3     LDB  DP3    PAREN ERROR -- EXTRA (
0521  04340 065351        JMP  BAD
0522*
0523  04341 025716        LDB  THIS   IS THIS A )?
0524  04342 014056        CPB  IRPR
0525  04343 066360        JMP  WRBK   YES
0526*
0527  04344 021716        LDA  THIS   ADD NEW CHARACTER TO STACK
0528  04345 070142        SAR  4
0529  04346 070204        SAL  12
0530  04347 041717        IOR  T1
0531  04350 055541 STSK   DSZ  OP     FIRST DECREMENT THE POINTER
0532  04351 131541        STA  OP,I   THEN STASH THE CHARACTER
0533  04352 021541        LDA  OP     NEXT CHECK FOR STACK OVERFLOW
0534  04353 070076        TCA
0535  04354 000536        ADA  OPLM   OPLM-OP<0 OR BAD
0536  04355 070113        SAP  *+2
0537  04356 065362        JMP  EXIT
0538  04357 065375        JMP  E9A
0539*
0540  04360 010503 WRBK   CPA  ALPR   IT WAS A )
0541  04361 066364        JMP  *+3
```

```
0542   04362  024400  E4      LDB  DP4     PAREN ERROR -- EXTRA)
0543   04363  065351          JMP  BAD
0544   04364  121541          LDA  OP,I
0545   04365  070604          SAL  4
0546   04366  070542          SAR  12
0547   04367  031601          STA  STAT
0548   04370  045541          ISZ  OP
0549   04371  065362          JMP  EXIT
0550*
0551   04372  000000          NOP          RESERVED FOR CHECKSUM WORD
0552*
0553*
0554*  PRIORITY TABLE FOR STACK ROUTINE
0555*
0556*                         STACK     COMPARE
0557*  SYMBOL                 PRIORITY  PRIORITY
0558*
0559*  ;                        0         1
0560*  ←                       11         3
0561*  (                        1        13
0562*  ,                        2         3
0563*  OR                       3         3
0564*  AND                      4         4
0565*  NOT                      5         5
0566*  RELATIONAL               6         6
0567*  +-                       7         7
0568*  */                       8         8
0569*  U-                       8         9
0570*  IMP*                    10        10
0571*  ↑                       11        11
0572*  FCN                     12        13
0573*  PROCEDURE                1         1
0574*
0575*
0576*       SEE PAGE 5 FOR A FURTHER DISCUSSION
0577*
0578*
0579*  IMPORTANT LOCATIONS:
0580*
0581*       SKEY -- CONTAINS CURRENT KEYCODE IN ASCII FORM
0582*
0583*       CODE -- CONTAINS CURRENT KEYCODES IN INTERNAL FORM
```

```
0585*
0586*  GTO/GSB
0587*
0588   04373  062422  GTO     JSM   CHKB    GTO AND GSB MUST BEGIN
0589   04374  020376          LDA   DP6     THE STATEMENT
0590   04375  031602          STA   G
0591   04376  061304          JSM   CP1     THEIR SYNTAX IS IDENTICAL
0592   04377  025545          LDB   PSP
0593   04400  074133          SBP   *+2,C   ADJUST PSP TO BEGIN A
0594   04401  074070          SIB   *+1     MACHINE WORD
0595   04402  070742          SAR   16      LEAVE A BLANK WORD
0596   04403  074557          STA   B,I     FOR THE FAST-ACCESS ADDRESS
0597   04404  074070          SIB   *+1
0598   04405  035545          STB   PSP
0599   04406  065362          JMP   EXIT
0600*
0601*  G=0 GO #                             GENERAL PROCEDURE:
0602*
0603   04407  014366  GE9     CPB   CNFD    FIND OUT EXACTLY WHAT KIND
0604   04410  065304          JMP   CP1     OF GTO OR GSB THIS IS BY
0605   04411  014367          CPB   CNFZ    LOOKING AT THE FIRST CHARACTER
0606   04412  065304          JMP   CP1
0607*
0608*  G=8 GO END
0609*
0610   04413  014454  GE8     CP8   CNFE
0611   04414  066232          JMP   SNLL    TO BE TERMINATED BY A NULL
0612   04415  014457          CPB   CNFY    CHARACTER SO THAT THE REVERSE
0613   04416  066232          JMP   SNLL    COMPILER CAN SKIP OVER THE
0614   04417  065317          JMP   E1      LITERAL INFORMATION
0615*
0616*  BEGINS CHECK
0617*
0618   04420  010377  CHKB1   CPA   DP5
0619   04421  065274          JMP   DISIS   ALLOWS CODE TO BE USED FOR ENTER
                                               STMTS
0620   04422  010401  CHKB    CPA   DP3     CODE ALLOWED IF IT BEGINS A STMT
0621   04423  065274          JMP   DISIS
0622   04424  055777          DSZ   RSTAK   IF THERE IS AN ERROR, THE JSM
0623   04425  065317          JMP   E1      TO THIS SECTION MUST BE CHUCKED
0624*
0625*  THE FAST-ACCESS WORDS ARE SET BY THE GTO/GSB ROUTINES
0626*  IN THE COURSE OF NORMAL EXECUTION
0627*
0628*  HENCE EACH GTO/GSB IS EXECUTED SLOWLY ONCE AND
0629*  ONLY ONCE
0630*
0631*  THE FAST-ACCESS WORDS ARE RESET WHENEVER THE PROGRAM
0632*  COULD BE MODIFIED
```

```
0634*
0635*  QUOTE
0636*
0637   04426  010401  QUO    CPA  DP3      DOES THE " BEGIN A STATEMENT?
0638   04427  066444         JMP  ROU      YES, IT IS OK
0639   04430  010403         CPA  DP1      OTHERWISE CHECK AGAINST
0640   04431  066435         JMP  CHQ      THE SYNTAX TABLE
0641   04432  010400         CPA  DP4
0642   04433  066435         JMP  CHQ      CPA HAS THE WRONG SENSE
0643   04434  065317         JMP  E1
0644*
0645   04435  021545  CHQ    LDA  PSP
0646   04436  070152         SAM  *+3      IS CHARACTER TO BE STORED IN LEFT-HALF?
0647   04437  020510         LDA  SQT      YES,MAKE SKEY SPECIAL "
0648   04440  031544         STA  SKEY
0649   04441  021601         LDA  STAT
0650   04442  070042         SAR  2        HAVE TO CHECK STAT
0651   04443  071451         SLA  *-7
0652*
0653   04444  020400  ROU    LDA  DP4      THE QUOTE IS DEFINITELY OK
0654   04445  031602         STA  G        SO SET G TO DO A LITERAL TRANSFER
0655   04446  061274         JSM  DISIS    THERE IS NO IMPLIED STORAGE
0656   04447  065304         JMP  CP1      OF A QUOTE STRING
0657*
0658*  G=5 END QUOTE
0659*
0660   04450  070742  GE5    SAR  16       AT THE CLOSING " RESET G
0661   04451  031602         STA  G
0662   04452  031606         STA  PS
0663   04453  014451         CPB  CNFC     THE STRING MUST BE FOLLOWED BY
0664   04454  066071         JMP  CMPLR    A COMMA OR THE END OF THE
0665   04455  014454         CPB  CNFE     STATEMENT OR LINE
0666   04456  066071         JMP  CMPLR
0667   04457  014457         CPB  CNFY
0668   04460  066071         JMP  CMPLR
0669   04461  065317         JMP  E1
0670*
0671*  THERE ARE TWO MAIN USES FOR QUOTES:
0672*
0673*  1)  AS A LABEL FOR A LINE
0674*
0675*  2)  AS ALPHAMERIC INFORMATION FOR AN OUTPUT ROUTINE
0676*      SUCH AS PRINT OR DISPLAY
0677*
0678*  A LABEL FOR A LINE IS "INTERPRETED" BY BUILDING THE
0679*  ALPHAMERIC CODES IN THE RUN-TIME STACK
0680*
0681*  THIS IS A WASTE OF TIME, BUT OTHERWISE DOES NO HARM
0682*
0683*  THE GTO AND GSB ROUTINES SCAN THE BEGINNING OF EACH
0684*  LINE IN SEARCH OF A LABEL, WHEN NECESSARY FOR ALPHA
0685*  GTO OR GSB
```

```
0687*
0688*  NUMERIC
0689*
0690   04472 072150 NUM    RZA *+3
0691   04473 020402        LDA DP2
0692   04474 066065        JMP CM
0693   04475 023724        LDA ADPS
0694   04476 170000        CLR             CLEAR THE WORK AREAS
0695   04477 020402        LDA DP2         IN PREPARATION FOR CHECKING
0696   04470 031602        STA G
0697   04471 020511        LDA ALP1        STORE AN ALPHA CHARACTER TO SIGNAL
0698   04472 061305        JSM CP2         THE BEGINNING OF THIS NUMBER
0699*
0700*  G=2 ANALYZE NUMERIC CHARACTER
0701*
0702   04473 025540        LDB CN          DOES IT START WITH A DIGIT OR DEC?
0703   04474 014366 GE2    CPB CNFD        FIND OUT BY LOOKING AT THE
0704   04475 066521        JMP CD          CONTROL NUMBER
0705   04476 014367        CPB CNFZ
0706   04477 066521        JMP CD
0707   04500 014365        CPB CNFP
0708   04501 066524        JMP CDP
0709   04502 014364        CPB CNFX        OR DOES IT START WITH AN EEX?
0710   04503 066531        JMP CE
0711*
0712   04504 021610        LDA SD          IT WAS NONE OF THESE, SO IT MUST BE
0713   04505 070110        SZA *+2         THE END OF THE NUMBER STRING
0714   04506 066236        JMP STZG        OR THE + OR – EXPONENT SIGNS
0715*
0716   04507 021607        LDA DP          WRAP IT UP
0717   04510 070056        CMA
0718   04511 051611        AND ES          WERE THERE ANY MANTISSA DIGITS?
0719   04512 070310        SZA *+6         NO, JUST AN ISOLATED DEC
0720   04513 045607        ISZ DP
0721   04514 014374        CPB CNFS        STORE THE – EXPONENT SIGN
0722   04515 065304        JMP CP1
0723   04516 014375        CPB CNFA        SCRAP THE + EXPONENT SIGN
0724   04517 065362        JMP EXIT
0725   04520 065317        JMP E1
0726*
0727   04521 020403 CD     LDA DP1         INDICATE AT LEAST ONE MANTISSA DIGIT
0728   04522 031610        STA SD          WAS ENTERED, THEN STORE THAT DIGIT
0729   04523 065304        JMP CP1
0730*
0731   04524 021607 CDP    LDA DP          CHECK FOR MORE THAN ONE DEC
0732   04525 041611        IOR ES
0733   04526 073510        RZA *–6
0734*
0735   04527 045607        ISZ DP          IF ONLY ONE DEC, STORE IT
0736   04530 065304        JMP CP1
0737*
0738   04531 021610 CE     LDA SD          PROCESS THE EXPONENT SPECIFICATION
0739   04532 070056        CMA
0740   04533 051607        AND DP
0741   04534 041611        IOR ES
0742   04535 073150        RZA *–3         ILLEGAL EXPONENT SPECIFICATION
```

```
0743*
0744    04536  045611          ISZ   ES
0745    04537  070742          SAR   16
0746    04540  031607          STA   DP
0747    04541  025610          LDB   SD
0748    04542  031610          STA   SD
0749    04543  076150          RZB   *+3      DOES THE EXPONENT SPECIFICATION STAND
0750    04544  020504          LDA   AONE     ALONE OR WERE THERE MANTISSA DIGITS?
0751    04545  061305          JSM   CP2      IT STANDS ALONE
0752    04546  065304          JMP   CP1
0753*
0754*   THE NUMERIC ROUTINE DOES NO CONVERSION TO INTERNAL FORMAT
0755*
0756*   ITS MAIN FUNCTIONS ARE:
0757*
0758*   1)  CHECK FOR PROPER NUMBER SYNTAX
0759*
0760*   2)  CHANGE . . . EEX TO . . . 1 EEX
0761*
0762*       EXPONENT OVERFLOW/UNDERFLOW IS CHECKED AT RUN TIME
0763*
0764*   SINCE ADJACENT NUMBERS MAY APPEAR IN THE REVERSE POLISH
0765*   OUTPUT STRING, A SPECIAL CHARACTER (ALPHA CHARACTER) IS
0766*   USED TO SEPARATE NUMBERS
```

```
0768*  INTERPRETER BEGINS HERE
0769*
0770*  INTERPRETS THE POLISH NOTATION PRODUCED BY
0771*  THE COMPILER
0772*
0773*  THE INTERPRETER CONSISTS OF A CONTROL ROUTINE, DESIGNED
0774*  TO FETCH CODES FROM THE PROGRAM AND ROUTE CONTROL
0775*  TO THE APPROPRIATE RUN-TIME ROUTINE
0776*
0777*  AND THE RUN-TIME ROUTINES THEMSELVES
0778*
0779*  SOME OF THE RUN-TIME ROUTINES ARE THE MATH ROUTINES
0780*  WHICH ARE IN A SEPARATE LISTING
0781*
0782*  THE INDIVIDUAL RUN-TIME ROUTINES
0783*  ARE NUMEROUS BUT STRAIGHTFORWARD
0784*
0785*  THE INTERPRETER MAINTAINS A STACK TO MANAGE
0786*  OPERAND ADDRESSES AND TEMPORARY RESULTS
0787*
0788*     ********
0789*     *      *
0790*     *      *
0791*     *      *
0792*     *      *   ← TP  TEMPORARY RESULT POINTER
0793*     *      *
0794*     *      *
0795*     *      *   TEMPORARY RESULTS ARE IN LOW-CORE END
0796*     *      *
0797*     *      *
0798*     .
0799*     .
0800*     .
0801*
0802*     *      *
0803*     *      *
0804*     *      *   ADDRESSES ARE IN HIGH-CORE END
0805*     *      *
0806*     *      *
0807*     *      *   ← AP1  TOP OF ADDRESS STACK
0808*     *      *
0809*     *      *
0810*     *      *   ← AP2  CURRENT RETURN ADDRESS
0811*     *      *
0812*     *      *
0813*     *      *   ← BOTTOM CONTAINS A -1
0814*     ********
```

```
0816*  AP1 POINTS TO THE CURRENTLY VALID TOP ADDRESS
0817*
0818*  THE AP1 SIGN BIT SIGNIFIES:
0819*
0820*  IF CLEAR — THE ADDRESS IS A PERMANENT REGISTER
0821*
0822*  IF SET   — THE ADDRESS IS A TEMPORARY RESULT IN
0823*             THE UPPER STACK
0824*
0825*
0826*
0827*  AP2 POINTS TO LINK INFORMATION FOR THE LAST-EXECUTED
0828*  GSB
0829*
0830*  THE ENTIRE SET OF LINK WORDS IS STORED IN A
0831*  CHAINED LIST TERMINATED BY THE -1 AT THE BOTTTOM OF
0832*  THE STACK
0833*
0834*
0835*
0836*  WHEN A GSB IS EXECUTED A REMINDER FLAG (GTS) IS SET
0837*
0838*  THEN AT THE END OF THE LINE THE RETURN INFORMATION
0839*  IS ACTUALLY STORED
0840*
0841*  THIS IS DONE MAINLY BECAUSE A GTO COULD CANCEL A
0842*  PREVIOUS GSB
0843*
0844*  (CONSIDER ALTERNATING IF AND GTO OR GSB STATEMENTS)
0845*
0846*
0847*
0848*  THE FOLLOWING KEYCODES ARE EXECUTED IN ROUTINES
0849*  EXTERNAL TO THE INTERPRETER:
0850*
0851*       ALL MATH
0852*       ENTER
0853*       STOP
0854*       NORMAL
0855*       TRACE
0856*       FIXED
0857*       FLOAT
0858*       SPACE
0859*       PRINT
0860*       DISPLAY
0861*       LOAD
0862*       RECORD
0863*       END
0864*       ALL OPTION BLOCK CODES
```

```
0866*
0867*  INTERPRETER SETUP (MAIN ENTRY)
0868*
0869   04547  021542 INTRP   LDA ENDS    FIRST SET UP THE REQUIRED POINTERS
0870   04550  000407         ADA DM3     FOR THE RUN-TIME STACK
0871   04551  031614         STA TPP
0872   04522  001615         ADA BSTK
0873   04553  000401         ADA DP3
0874   04554  031557         STA AP2
0875   04555  020405         LDA DM1
0876   04556  131557         STA AP2,I
0877   04557  170402         RET
0878   04560  035605 XRH      STB IMA     SAVE THE STARTING PROGRAM ADDRESS
0879*
0880*  ;
0881*
0882   04561  070742 XRI      SAR 16      RESET THINGS IN PREPARATION FOR A
0883   04562  031604         STA NMA     NEW LINE
0884   04563  031702         STA GTS
0885   04564  025614 XR       LDB TPP     RESET THINGS IN PREPARATION FOR A
0886   04565  035551         STB TP      NEW STATEMENT
0887   04566  025557         LDB AP2
0888   04567  035556         STB AP1
0889*
0890*  GO TO EXECUTE ROUTINE
0891*
0892   04570  062575 XJ       JSM SXJ     LOOK UP THE NEXT CODE TO BE EXECUTED
0893   04571  003701         ADA TAB3    AND BRANCH TO AN EXECUTION ROUTINE
0894   04572  070517         LDA A,I     VIA TABLE TAB3
0895   04573  070016         EXA
0896   04574  066570         JMP XJ
0897*
0898*  PICK UP NEXT CODE (SUBROUTINE)
0899*
0900   04575  125605 SXJ      LDB IMA,I   THIS SUBROUTINE GETS THE NEXT CODE
0901   04576  074173         SBP *+3,C   FROM THE MEMORY AREA DESIGNATED BY
0902*                                     IMA AND EXTRACTS PERTINENT
0903   04577  074517         LDA B,I     FIELDS OF INFORMATION FOR LATER USE
0904   04600  076230         RIB *+4
0905*
0906   04601  074517         LDA B,I
0907   04602  070302         SAR 7
0908   04603  076053         SBP *+1,S
0909*
0910   04604  135605         STB IMA,I   PERTINENT INFORMATION:
0911   04605  050345         AND RH
0912   04606  031544         STA SKEY    SETUP SKEY,CODE,AND CN
0913   04607  000377         ADA TADD
0914   04610  070517         LDA A,I
0915   04611  031561         STA CODE
0916   04612  070442         SAR 10
0917   04613  031540         STA CN
0918   04614  170402         RET
```

```
0920*
0921*  SETUPS
0922*
0923   04615  125556  XBSET   LDB  AP1,I
0924   04616  045556          ISZ  AP1
0925   04617  074473          SBP  *+9,C
0926*
0927   04620  035553          STB  OPND2     THIS ROUTINE UPDATES THE STACK
0928   04621  021551          LDA  TP        POINTERS IN PREPARATION FOR A
0929   04622  125556          LDB  AP1,I     BINARY ARITHMETIC OPERATION
0930   04623  074673          SBP  *+13,C
0931*
0932   04624  000410          ADA  DM4
0933   04625  031551          STA  TP
0934   04626  035555          STB  RESLT
0935   04627  066643          JMP  XU2
0936*
0937   04630  035553          STB  OPND2
0938*
0939   04631  125556  XUSET   LDB  AP1,I     UNARY OPERATOR
0940   04632  074173          SBP  *+3,C
0941*
0942   04633  035555          STB  RESLT
0943   04634  066643          JMP  *+7
0944*
0945   04635  021551          LDA  TP        THIS ROUTINE UPDATES THE STACK
0946   04636  000400          ADA  DP4       POINTERS IF REQUIRED FOR A
0947   04637  031551          STA  TP        UNARY ARITHMETIC OPERATION
0948   04640  031555          STA  RESLT
0949   04641  072053          SAP  *+1,S
0950   04642  131556          STA  AP1,I
0951*
0952   04643  035554  XU2     STB  OPND1
0953   04644  025540          LDB  CN
0954   04645  004514          ADB  ARITH
0955   04646  074737          JMP  B,I
0956*
0957*  OPERAND
0958*
0959   04647  055556  XOPD    DSZ  AP1       THIS ROUTINE BRINGS AN OPERAND
0960   04650  021561          LDA  CODE      TO THE RUN-TIME STACK
0961   04651  050345          AND  RH
0962   04652  000442          ADA  ADOP
0963   04653  131556          STA  AP1,I
0964*
0965   04654  025556  OVLP    LDB  AP1       MAKE SURE NO OVERLAP OCCURS
0966   04655  074056          CMB            BETWEEN THE TWO STACKS
0967   04656  005551          ADB  TP
0968   04657  004374          ADB  DP8
0969   04660  074113          SBP  *+2
0970   04661  066570          JMP  XJ        ALL OK, SO GET NEXT CODE
0971*
0972   04662  024373  E9B     LDB  DP9       MEMORY OVERFLOW
0973   04663  066705          JMP  EXER
```

```
0975*
0976*  GAZINTA
0977*
0978   04664  125556 XGAZ   LDB  AP1,I    THIS ROUTINE STORES AN ARITHMETIC
0979   04665  074153        SBP  *+3      VALUE IN A PERMANENT REGISTER (ONLY)
0980   04666  024376 E6     LDB  DP6      ILLEGAL STORE
0981   04667  066705        JMP  EXER
0982*
0983   04670  035550        STB  .STPT    RECORD THE STORE FOR TRACE
0984   04671  045556        ISZ  AP1      OR DISPLAY PURPOSES
0985   04672  121556 XGAZ1  LDA  AP1,I
0986   04673  070073        SAP  *+1,C
0987   04674  170004        XFR           DO THE TRANSFER
0988*
0989   04675  021705        LDA  .WMOD    IS TRACE REQUIRED?
0990   04676  070151        SLA  *+3
0991   04677  160310        JSM  .FMT2,I  YES
0992   04700  160250        JSM  .PRT1,I
0993   04701  066570        JMP  XJ
0994*
0995*  R-REGISTER
0996*
0997   04702  061321 SUBSC  JSM  FIXPT    GET ADDRESS OF AN R-REGISTER
0998   04703  074213        SBP  *+4      BY FIRST CONVERTING TO INTEGER
0999   04704  024377 E5     LDB  DP5      SUBSCRIPT OUT OF RANGE
1000*
1001   04705  061351 EXER   JSM  BAD
1002   04706  066570        JMP  XJ       ERROR DURING EXECUTION
1003*
1004   04707  074704        SBL  2        NOW CALCULATE AN ACTUAL CORE
                                           ADDRESS
1005   04710  004401        ADB  DP3
1006   04711  074056        CMB
1007   04712  005712        ADB  RR
1008*
1009   04713  021542        LDA  ENDS     CHECK MEMORY PROTECT BOUNDARY
1010   04714  001615        ADA  BSTK
1011   04715  070056        CMA
1012   04716  074017        ADA  B
1013   04717  071252        SAM  *-11     SKIP IF OUT OF RANGE
1014*
1015   04720  135556        STB  AP1,I    FINALLY PLACE ADDRESS IN STACK
1016   04721  066570        JMP  XJ
```

```
1018*
1019*  QUOTE
1020*
1021   04722  024405  XQU0  LDB  DM1     THIS ROUTINE PLACES ALPHAMERIC
1022   04723  035613        STB  N       INFORMATION IN THE TEMPORARY
1023   04724  055556        DSZ  AP1     STACK FOR USE BY OUTPUT ROUTINES
1024   04725  021551        LDA  TP
1025   04726  000400        ADA  DP4     THE INFORMATION IS PACKED TWO
1026   04727  031612        STA  M       CHARACTERS PER MACHINE WORD
1027   04730  040262        IOR  ALPH
1028   04731  131556        STA  AP1,I
1029*
1030   04732  062575  XGE4  JSM  SXJ     LOOK OUT FOR THE ENDING QUOTE
1031   04733  010452        CPA  CNFQ
1032   04734  066751        JMP  XENQ
1033*
1034   04735  055613        DSZ  N       NO ENDING QUOTE YET, SO PLACE NEXT
1035   04736  021544        LDA  SKEY    CHARACTER IN STACK
1036   04737  025612        LDB  M
1037   04740  074233        SBP  *+4,C   STACK THE CHARACTERS IN PACKED
1038   04741  074617        IOR  B,I     FORMAT
1039   04742  074557        STA  B,I
1040   04743  076230        RIB  *+4     COUNTER N IS THE CHARACTER COUNT
1041   04744  070404        SAL  8       IN 1'S COMPLEMENT FORM
1042   04745  074557        STA  B,I
1043   04746  076053        SBP  *+1,S
1044   04747  035612        STB  M
1045   04750  066732        JMP  XGE4    LOOP BACK TO GET NEXT CHARACTER
1046*
1047   04751  025612  XENQ  LDB  M       ENDING QUOTE FOUND
1048   04752  074273        SBP  *+5,C
1049   04753  074517        LDA  B,I
1050   04754  040360        IOR  ABLK    PAD THE STRING WITH A BLANK IF
1051   04755  074557        STA  B,I     REQUIRED TO ELIMINATE GARBAGE
1052   04756  074070        SIB  *+1
1053*
1054   04757  021613        LDA  N       READJUST ALL POINTERS AND GO
1055   04760  074557        STA  B,I     CHECK FOR OVERLAP
1056   04761  004407        ADB  DM3
1057   04762  035551        STB  TP
1058   04763  066654        JMP  OVLP
```

```
1060*
1061*  SPACE, FIXED, FLOAT
1062*
1063   04764  020305  FIX   LDA  FXD     THESE ROUTINES GET THE PARAMETER
1064   04765  066767        JMP  *+2     FOR FIXED, FLOAT, AND SPACE
1065*
1066   04766  020304  FLO   LDA  FLT
1067   04767  031720        STA  T2
1068*
1069   04770  025572        LDB  NRSLT
1070   04771  035550        STB  .STPT
1071   04772  024405        LDB  DM1     THE ACTUAL OPERATION OF THE ROUTINES
1072   04773  021556        LDA  AP1     IS DESCRIBED IN THEIR RESPECTIVE
1073   04774  011557        CPA  AP2     LISTINGS
1074   04775  165720        JMP  T2,I
1075*
1076   04776  061321        JSM  FIXPT   IN GENERAL, THE PARAMETER IS FETCHED
1077   04777  074117        LDA  B       AND CONVERTED TO FIXED-POINT
1078   05000  070212        SAM  *+4
1079   05001  000415        ADA  DM10    IF NO PARAMETER IS SUPPLIED, A
1080   05002  070113        SAP  *+2     SUBSTITUTE IS FURNISHED BY DEFAULT
1081*
1082   05003  165720        JMP  T2,I
1083*
1084   05004  024402  E2    LDB  DP2     ILLEGAL PARAMETER
1085   05005  065351        JMP  BAD
1086*
1087   05006  020372  SPA   LDA  DP10
1088   05007  025556        LDB  AP1
1089   05010  015557        CPB  AP2
1090   05011  164473        JMP  SPC,I
1091*
1092   05012  061321        JSM  FIXPT
1093   05013  074117        LDA  B
1094   05014  071412        SAM  E2
1095   05015  070410        SZA  *+8
1096   05016  000420        ADA  DM16
1097   05017  071253        SAP  E2
1098   05020  074744        SBL  1
1099   05021  074117        LDA  B
1100   05022  074704        SBL  2
1101   05023  074017        ADA  B
1102   05024  164473        JMP  SPC,I
1103*
1104   05025  170402        RET
```

```
1106*
1107*  IF
1108*
1109   05026  061321  XIF     JSM  FIXPT     IF WILL SKIP TO NEXT STATEMENT OR NOT
1110   05027  074150          SZB  *+3          DEPENDING ON WHETHER ITS PARAMETER
1111   05030  066570          JMP  XJ        IS ZERO OR NONZERO RESPECTIVELY
1112*
1113   05031  035604          STB  NMA
1114*
1115   05032  121605          LDA  IMA,I     SKIP TO END OF LINE
1116   05033  070073          SAP  *+1,C
1117   05034  070070          SIA  *+1
1118   05035  070537          LDB  A,I
1119   05036  075713          SBP  *-2
1120   05037  131605          STA  IMA,I
1121   05040  067047          JMP  XEOL
1122*
1123*  JUMP
1124*
1125   05041  061321  XJMP    JSM  FIXPT     JUMP WILL SKIP A VARIABLE NUMBER
1126   05042  035613  MOVE    STB  N            OF STATEMENTS ACCORDING TO ITS
1127   05043  125605          LDB  IMA,I     PARAMETER
1128   05044  063472          JSM  RELG
1129   05045  075213          SBP  *-12      DID WE SKIP OUT OF RANGE?
1130   05046  065357          JMP  BAD+6        YES
1131*
1132*  EOL
1133*
1134   05047  021705  XEOL    LDA  .WMOD     FIRST DETERMINE IF ANY ERRORS
1135   05050  050512          AND  ERM1
1136   05051  073650          RZA  *-3
1137   05052  021702          LDA  GTS       NEXT DETERMINE IF A GSB MUST BE
1138   05053  070110          SZA  *+2          COMPLETED
1139   05054  067164          JMP  XGSS
1140   05055  021604  XRR     LDA  NMA       NEXT DETERMINE IF A PREVIOUS
1141   05056  072210          RZA  *+4          GTO OR GSB CAUSED A TRANSFER
1142   05057  121605          LDA  IMA,I        INSURE THAT NEXT LINE IS BEGUN
1143   05060  070133          SAP  *+2,C        PROPERLY
1144   05061  070070          SIA  *+1
1145   05062  131605          STA  IMA,I
1146*
1147   05063  021705          LDA  .WMOD     NEXT DETERMINE IF TRACE MODE IS SET
1148   05064  072110          RZA  *+2
1149   05065  066561          JMP  XRI       NO TRACE SO CONTINUE
1150*
1151   05066  065373          JMP  EXIT+9
1152*
1153*  ABORT EXECUTION
1154*
1155   05067  161710  NONX    JSM  XEND,I    PROGRAM HAS NO END!
1156   05070  067047          JMP  XEOL
```

```
1158*
1159*  PRINT
1160*
1161   05071  020250  XPRNT  LDA  .PRT1    ENTER HERE FOR PRINT
1162   05072  067074         JMP  *+2
1163*
1164*  DISPLAY
1165*
1166   05073  023725  XDISP  LDA  ADSP     ENTER HERE FOR DISPLAY
1167*
1168   05074  031611         STA  ES       SAVE ADDRESS OF PRINT OR DISPLAY
1169*
1170*  GET MIXED PARAMETERS
1171*
1172   05075  021557         LDA  AP2      THESE ROUTINES GET THE ACTUAL
1173   05076  031612         STA  M        PARAMETERS FOR PRINT AND DISPLAY
1174   05077  011556         CPA  AP1
1175   05100  067110         JMP  GTR      SKIP IF NO PARAMETER SUPPLIED
1176*
1177   05101  020405         LDA  DM1      FIRST DETERMINE WHAT TYPE OF
                                            PARAMETER
1178   05102  031550         STA  .STPT
1179   05103  055612  GMORE  DSZ  M        M POINTS TO NEXT PARAMETER
1180   05104  121612         LDA  M,I
1181   05105  031550         STA  .STPT    SAVE ADDRESS FOR LATER DISPLAY
1182   05106  070744         SAL  1
1183   05107  070732         SAM  GALPH,C IF — ALPHAMERIC
1184*
1185   05110  160310  GTR    JSM  .FMT2,I
1186   05111  161611         JSM  ES,I
1187   05112  021705         LDA  .WMOD
1188   05113  070313         SAP  *+6
1189   05114  021563         LDA  TSW      NEW KEY ENTERED
1190   05115  000410         ADA  DM4
1191   05116  072250         RZA  *+5
1192   05117  160525         JSM  CKSTP,I  PROGRAM MODE,PROCESS KEY IF
                                            INTERRUPTABLE
1193   05120  070152         SAM  *+3      – IF STOP KEY
1194   05121  021612         LDA  M        DONE WITH THIS PARAMETER
1195   05122  011556         CPA  AP1      PARAMETER LIST EXHAUSTED?
1196   05123  066570         JMP  XJ       YES
1197   05124  067103         JMP  GMORE    NO
1198*
1199*
1200   05125  070002  GALPH  SAR  1        THIS ROUTINE MOVES ALPHAMERIC
                                            INFO TO .IBUF
1201   05126  031717         STA  T1       T1 = SOURCE
1202   05127  020554         LDA  .IBUF
1203   05130  031720         STA  T2       T2 = DESTINATION
1204   05131  160255         JSM  L18A,I   CLEAR BUFFER
1205   05132  024516         LD8  DM8
1206   05133  121717         LDA  T1,I     DO THE TRANSFER
1207   05134  070252         SAM  *+5
1208   05135  131720         STA  T2,I
1209   05136  045717         ISZ  T1
1210   05137  045720         ISZ  T2
1211   05140  077570         RIB  *–5
1212   05141  067110         JMP  GTR
```

```
1214*
1215*  DISPLAY ROUTINE
1216*
1217   05142  020516  RDSPY   LDA  DM8      THIS ROUTINE CALLS THE DISPLAY
1218   05143  031607          STA  OP       ROUTINE (ALSO WATCH FOR NEW KEY)
1219   05144  020207  RDS1    LDA  .DBUF
1220   05145  031721          STA  .L
1221   05146  160206          JSM  .DSP1,I
1222   05147  045607          ISZ  DP       NO SO STAY IN DISPLAY LOOP
1223   05150  067144          JMP  RDS1
1224   05151  170402          RET
```

(142)

```
1226*
1227*  RETURN
1228*
1229   05152  125557  XRET    LDB  AP2,I    THIS ROUTINE OBTAINS THE RETURN
1230   05153  014405            CPB  DM1     ADDRESS FROM THE RUN-TIME STACK
1231   05154  067162            JMP  E7
1232   05155  045557            ISZ  AP2
1233   05156  121557            LDA  AP2,I
1234   05157  031604            STA  NMA     THE R.A.
1235   05160  035557            STB  AP2
1236   05161  066570            JMP  XJ
1237*
1238   05162  024375  E7        LDB  DP7     RET HAS NO MATCHING GSB
1239   05163  066705            JMP  EXER
1240*
1241*  GSB SETUP
1242*
1243   05164  020333  XGSS      LDA  ASTEN   THIS ROUTINE STASHES THE RETURN
1244   05165  025563            LDB  TSW     ADDRESS INTO THE RUN-TIME STACK
1245   05166  014400            CPB  DP4
1246   05167  121605            LDA  IMA,I   THIS IS DONE AFTER EOL IS ENCOUNTERED
1247   05170  070133            SAP  *+2,C
1248   05171  070070            SIA  *+1
1249*
1250   05172  025556  XGSS2     LDB  AP1     A "FAKE" RETURN ADDRESS IS SUPPLIED
1251   05173  004405            ADB  DM1     IF GSB WAS EXECUTED FROM KEYBOARD
1252   05174  074557            STA  B,I
1253   05175  004405            ADB  DM1
1254   05176  021557            LDA  AP2
1255   05177  035556  XGSS1     STB  AP1
1256   05200  035557            STB  AP2
1257   05201  131557            STA  AP2,I
1258*
1259   05202  074056            CMB          CHECK FOR OVERLAP
1260   05203  005551            ADB  TP
1261   05204  004400            ADB  DP4
1262   05205  074113            SBP  *+2
1263   05206  067055            JMP  XRR     ALL OK
1264   05207  066662            JMP  E9B
```

```
1266*
1267*  GOTO
1268*
1269   05210  061253  XGSB    JSM  XNSTP
1270   05211  020403          LDA  DP1
1271   05212  067214          JMP  *+2      ACCORDING TO WHETHER GTO OR GSB
1272   05213  070742  XGTO    SAR  16
1273   05214  031702          STA  GTS
1274   05215  121605          LDA  IMA,I    NEXT CHECK FAST-TRANSFER WORD
1275   05216  070133          SAP  *+2,C
1276   05217  070070          SIA  *+1
1277   05220  025567          LDB  MPH      IF MPH=0 FAST GTO MODE OKAY;ELSE NOT
1278   05221  076550          RZB  *+11
1279   05222  070537          LDB  A,I
1280   05223  074450          SZB  *+9
1281   05224  035604          STB  NMA      FAST TRANSFER BEGINS HERE
1282   05225  035546          STB  CMA      BY UPDATING NMA, CMA, & IMA
1283   05226  070070          SIA  *+1
1284   05227  131605          STA  IMA,I
1285   05230  062575          JSM  SXJ
1286   05231  010356          CPA  CNFN     SKIP ALONG TO END OF STATEMENT
1287   05232  066570          JMP  XJ       FOUND IT
1288   05233  067230          JMP  *-3
1289*
1290   05234  031717          STA  T1       SLOW GTO BEGINS HERE
1291   05235  070070          SIA  *+1
1292   05236  131605          STA  IMA,I
1293   05237  062575          JSM  SXJ      LOOK AT NEXT CODE TO DETERMINE TYPE
1294   05240  074742          SBR  16       OF GTO OR GSB
1295   05241  010452          CPA  CNFQ
1296   05242  067313          JMP  QGTO     IT IS ALPHAMERIC
1297   05243  010351          CPA  CNFO
1298   05244  165623          JMP  GOTF,I   IT IS TO AN OPTION BLOCK
1299   05245  035613          STB  N
1300   05246  027722          LDB  TAB5     IT IS NUMERIC
1301   05247  035720          STB  T2
1302   05250  010375          CPA  CNFA     IS IT RELATIVE OR ABSOLUTE?
1303   05251  067255          JMP  GA
1304   05252  010374          CPA  CNFS
1305   05253  067256          JMP  GS
1306   05254  067262          JMP  XG4
```

(144)

```
1308*
1309  05255  045720  GA     ISZ  T2     IT IS RELATIVE
1310  05256  045720  GS     ISZ  T2
1311*
1312  05257  062575  XGES   JSM  SXJ    EXTRACT THE DIGITS
1313  05260  010356         CPA  CNFN
1314  05261  165720         JMP  T2,I
1315*
1316  05262  025613  XG4    LDB  N      CONVERT TO A DISPLACEMENT
1317  05263  074744         SBL  1      BY MULTIPLYING PREVIOUS ANSWER BY 10
1318  05264  035612         STB  M      AND ADDING IN NEXT DIGIT OF STATEMENT #
1319  05265  074704         SBL  2
1320  05266  005612         ADB  M
1321  05267  021561         LDA  CODE
1322  05270  050365         AND  DP15
1323  05271  070037         ADB  A
1324  05272  035613         STB  N      SAVE NEW DISPLACEMENT
1325  05273  067257         JMP  XGES
```

```
1327*
1328*  ABSOLUTE GOTO
1329*
1330   05274  063526  GABS   JSM  GTOP     FIRST GO TO TOP OR PROGRAM
1331   05275  063472  XG1    JSM  RELG     THEN DO A RELATIVE GTO
1332   05276  074212  XG2    SBM  *+4
1333   05277  035546         STB  CMA
1334   05300  035604         STB  NMA
1335   05301  135717         STB  T1,I
1336   05302  066570         JMP  XJ
1337*
1338*  SKIP BACKWARD
1339*
1340   05303  021613  GMNS   LDA  N        IT IS A RELATIVE — GTO
1341   05304  070076         TCA           SO CHANGE SIGN OF DISPLACEMENT
1342   05305  031613         STA  N
1343*
1344*  SKIP FORWARD
1345*
1346   05306  125605  GPLS   LDB  IMA,I    IT IS A RELATIVE + GTO
1347   05307  063472         JSM  RELG
1348   05310  075512         SBM  GMNS-1
1349   05311  135717         STB  T1,I
1350   05312  067300         JMP  XG2+2
1351*
1352*  ALPHA GOTO
1353*
1354   05313  035721  QGT0   STB  H1       IT IS AN ALPHAMERIC GTO
1355   05314  035722         STB  H2
1356*
1357   05315  062575  XGE1   JSM  SXJ
1358   05316  010452         CPA  CNFQ
1359   05317  067333         JMP  GQ1
1360*
1361   05320  021721         LDA  H1       BUILD THE COMPARISON WORD
1362   05321  070404         SAL  8        IN H1 AND H2
1363   05322  025722         LDB  H2
1364   05323  074342         SBR  8
1365   05324  074217         IOR  B
1366   05325  031721         STA  H1
1367   05326  021722         LDA  H2
1368   05327  070404         SAL  8
1369   05330  041544         IOR  SKEY
1370   05331  031722         STA  H2
1371   05332  067315         JMP  XGE1
1372*
1373   05333  063526  GQ1    JSM  GTOP     SEARCH PROGRAM FOR THIS LABEL
1374   05334  063401  GQ2    JSM  GAS2     FIRST FIND ANY LABEL WHATSOEVER
1375   05335  014405         CPB  DM1
1376   05336  066570         JMP  XJ
1377   05337  010517         CPA  BQT
1378   05340  067342         JMP  *+2
1379   05341  067334         JMP  *-5
1380*
1381   05342  074073  GQ4    SBP  *+1,C    A LABEL HAS BEEN FOUND
1382   05343  035720         STB  T2
```

```
1383  05344 074742          SBR  16
1384  05345 035723          STB  H3
1385  05346 035724          STB  H4
1386  05347 063400 GQ3      JSM  GAS1    EXTRACT THE LABEL CHARACTERS
1387  05350 010546          CPA  QUOTE
1388  05351 067336          JMP  MATCH   THE ENDING QUOTE HAS BEEN FOUND
1389  05352 031561          STA  CODE
1390  05353 021723          LDA  H3      BUILD THE ALPHAMERIC FIELD IN H3
                                            AND H4
1391  05354 070404          SAL  8
1392  05355 025724          LDB  H4
1393  05356 074342          SBR  8
1394  05357 074217          IOR  B
1395  05360 031723          STA  H3
1396  05361 021724          LDA  H4
1397  05362 070404          SAL  8
1398  05363 041561          IOR  CODE
1399  05364 031724          STA  H4
1400  05365 067347          JMP  GQ3
1401*
1402  05366 021721 MATCH     LDA  H1      CHECK AGAINST THE COMPARISON WORD
1403  05367 011723          CPA  H3
1404  05370 067372          JMP  *+2     ONLY THE LAST FOUR CHARACTERS WILL
1405  05371 067334          JMP  GQ2     BE SIGNIFICANT (ALLOWS FOR SUFFIX)
1406  05372 021722          LDA  H2
1407  05373 011724          CPA  H4
1408  05374 067376          JMP  *+2
1409  05375 067334          JMP  GQ2
1410  05376 025720          LDB  T2
1411  05377 067276          JMP  XG2
1412*
1413  05400 025612 GAS1      LDB  M       ROUTINE TO GET SYMBOLS FROM PROGRAM
1414  05401 074233 GAS2      SBP  *+4,C
1415*
1416  05402 074517          LDA  B,I     THIS ROUTINE IS SIMILAR TO THE BASIC
1417  05403 050345          AND  RH      ROUTINE SXJ
1418  05404 076330          RIB  *+6
1419*
1420  05405 074517          LDA  B,I
1421  05406 010405          CPA  DM1
1422  05407 067524          JMP  E8      JUMP IF LABEL CANNOT BE FOUND
1423  05410 070302          SAR  7
1424  05411 076053          SBP  *+1,S
1425*
1426  05412 035612          STB  M
1427  05413 170402          RET
```

```
1429*
1430*  FROM FLAG
1431*
1432   05414  063454 STFF JSM  BMSK    THESE ROUTINES RETURN THE VALUE
1433   05415  014377      CPB  DP5     OF A DESIGNATED FLAG
1434   05416  066705      JMP  EXER
1435*
1436   05417  074257      AND  B
1437   05420  070137      LDB  A
1438*
1439   05421  121556      LDA  AP1,I
1440   05422  070372      SAM  *+7,C
1441   05423  021551      LDA  TP
1442   05424  000400      ADA  DP4
1443   05425  031551      STA  TP
1444   05426  072053      SAP  *+1,S
1445   05427  131556      STA  AP1,I
1446   05430  070073      SAP  *+1,C
1447*
1448   05431  170000      CLR
1449   05432  074210      SZB  *+4
1450   05433  070070      SIA  *+1
1451   05434  024431      LDB  BCD1
1452   05435  070577      STB  A,I
1453   05436  066570      JMP  XJ
1454*
1455*  INTO FLAG
1456*
1457   05437  024405 STFI  LDB  DM1
1458   05440  067442      JMP  *+2
1459*
1460   05441  074742 CLFI  SBR  16
1461   05442  035720      STB  T2
1462*
1463   05443  063454      JSM  BMSK    THESE ROUTINES SET THE VALUE
1464   05444  014377      CPB  DP5     OF A DESIGNATED FLAG
1465   05445  066705      JMP  EXER
1466*
1467   05446  074217      IOR  B
1468   05447  074056      CMB
1469   05450  045720      ISZ  T2
1470   05451  074257      AND  B
1471   05452  031430      STA  FLAG
1472   05453  066570      JMP  XJ
```

```
1474*
1475*  BUILD MASK
1476*
1477   05454  061321 BMSK JSM  FIXPT   BUILD A MASK CONTAINING A "ONE" IN
1478   05455  074153       SBP  *+3    BIT POSITION N FOR FLAG N
1479   05456  024377       LDB  DP5
1480   05457  170402       RET
1481   05460  004420       ADB  DM16
1482   05461  075653       SBP  *-3
1483*
1484   05462  074056       CMB
1485   05463  074544       SBL  5
1486   05464  023717       LDA  FROT
1487   05465  074217       IOR  B
1488   05466  024403       LDB  DP1
1489   05467  070016       EXA
1490   05470  021430       LDA  FLAG
1491   05471  170402       RET
```

```
1493*
1494*  SKIPPING UTILITIES
1495*
1496   05472  074173 RELG    SBP  *+3,C     DO A RELATIVE GTO
1497   05473  067475          JMP  *+2
1498*
1499   05474  004405          ADB  DM1
1500   05475  074517          LDA  B,I
1501   05476  071713          SAP  *-2
1502   05477  010405          CPA  DM1
1503   05500  067524          JMP  E8
1504   05501  021613          LDA  N
1505   05502  070450          SZA  *+9
1506   05503  070513          SAP  *+10
1507*
1508   05504  004405          ADB  DM1
1509   05505  074517          LDA  B,I
1510   05506  071713          SAP  *-2
1511   05507  010405          CPA  DM1
1512   05510  067524          JMP  E8
1513   05511  045613          ISZ  N
1514   05512  067504          JMP  *-6
1515   05513  170402          RET
1516*
1517   05514  074517          LDA  B,I
1518   05515  070313          SAP  *+6
1519   05516  055613          DSZ  N
1520   05517  067521          JMP  *+2
1521   05520  170402          RET
1522   05521  010405          CPA  DM1
1523   05522  067524          JMP  E8
1524   05523  077470          RIB  *-7
1525*
1526   05524  024374 E8       LDB  DP8     GTO OR GSB OUT OF BOUNDS
1527   05525  065351          JMP  BAD
1528*
1529   05526  025546 GTOP     LDB  CMA     THIS ROUTINE FINDS THE TOP OF
1530   05527  074073          SBP  *+1,C   THE PROGRAM
1531   05530  004405          ADB  DM1
1532   05531  074517          LDA  B,I
1533   05532  074070          SIB  *+1
1534   05533  010405          CPA  DM1
1535   05534  170402          RET
1536   05535  004406          ADB  DM2
1537   05536  067531          JMP  *-5
```

```
1539*
1540*  BUILD NUMBER
1541*
1542  05537  020450  XNUM   LDA  ADR2    THIS ROUTINE CONVERTS A STRING OF
1543  05540  170000         CLR          DIGITS TO INTERNAL FORMAT
1544  05541  023724         LDA  ADPS
1545  05542  170000         CLR
1546  05543  024370         LDB  DP12
1547  05544  035613         STB  N
1548  05545  024405         LDB  DM1
1549  05546  035612         STB  M
1550*
1551  05547  062575  XGE2   JSM  SXJ      GET A DIGIT
1552  05550  010366         CPA  CNFD
1553  05551  067621         JMP  XD
1554  05552  010367         CPA  CNFZ
1555  05553  067641         JMP  XZ
1556  05554  010365         CPA  CNFP
1557  05555  067650         JMP  XDP
1558  05556  010364         CPA  CNFX
1559  05557  067652         JMP  XEE
1560  05560  021606  XF     LDA  PS
1561  05561  025611         LDB  ES
1562  05562  074110         SZB  *+2
1563  05563  070076         TCA
1564  05564  001612         ADA  M
1565  05565  070137         LDB  A
1566  05566  074113         SBP  *+2
1567  05567  074076         TCB
1568  05570  004550         ADB  DM100
1569  05571  074152         SBM  *+3
1570  05572  024372  E10    LDB  DP10     NUMERIC OVERFLOW
1571  05573  066705         JMP  EXER
1572*
1573  05574  171450         NRM          WRAP IT UP
1574  05575  070154         SES  *+3
1575  05576  070404         SAL  8
1576  05577  130450         STA  ADR2,I
1577  05600  020450         LDA  ADR2
1578  05601  025551  STNM   LDB  TP
1579  05602  004400         ADB  DP4
1580  05603  035551         STB  TP
1581  05604  170004         XFR
1582  05605  055556         DSZ  API
1583  05606  025551         LDB  TP
1584  05607  076053         SBP  *+1,S
1585  05610  135556         STB  API,I
1586  05611  025556         LDB  API
1587  05612  074056         CMB
1588  05613  005551         ADB  TP
1589  05614  004400         ADB  DP4
1590  05615  074153         SBP  *+3
1591  05616  021540  RECY   LDA  CN       RECYCLE THE CURRENT CODE
1592  05617  066571         JMP  XJ+1     WITHOUT PICKING UP A NEW ONE
1593  05620  066662         JMP  E9B
1594*
```

```
1595  05621  025607 XD     LDB DP
1596  05622  076150        RZB *+3
1597  05623  045612        ISZ M
1598  05624  067625        JMP *+1
1599  05625  045610        ISZ SD
1600  05626  025613        LDB N
1601  05627  076110        RZB *+2
1602  05630  067547        JMP XGE2
1603  05631  004405        ADB DM1
1604  05632  035613        STB N
1605  05633  171400        MLS
1606  05634  021561        LDA CODE
1607  05635  050365        AND DP15
1608  05636  140520        IOR ADRN,I
1609  05637  130520        STA ADRN,I
1610  05640  067547        JMP XGE2
1611*
1612  05641  025610 XZ     LDB SD
1613  05642  074110        SZB *+2
1614  05643  067621        JMP XD
1615  05644  025607        LDB DP
1616  05645  074110        SZB *+2
1617  05646  055612        DSZ M
1618  05647  067547        JMP XGE2
1619*
1620  05650  045607 XDP    ISZ DP
1621  05651  067547        JMP XGE2
1622*
1623  05652  074742 XEE    SBR 16
1624  05653  035610        STB SD
1625*
1626  05654  062575 XGE3   JSM SXJ
1627  05655  010366        CPA CNFD
1628  05656  067666        JMP XK
1629  05657  010367        CPA CNFZ
1630  05660  067666        JMP XK
1631*
1632  05661  025610        LDB SD
1633  05662  074110        SZB *+2
1634  05663  067560        JMP XF
1635*
1636  05664  045611        ISZ ES
1637  05665  067654        JMP XGE3
1638*
1639  05666  045610 XK     ISZ SD
1640  05667  025606        LDB PS
1641  05670  074744        SBL 1
1642  05671  035717        STB T1
1643  05672  074704        SBL 2
1644  05673  005717        ADB T1
1645  05674  021561        LDA CODE
1646  05675  050365        AND DP15
1647  05676  070037        ADB A
1648  05677  035606        STB PS
1649  05700  067654        JMP XGE3
```

ADDRESS TABLE FOR COMPLIER ROUTINES

```
1708*
1709*
1710   05767  004112  TTB1  DEF  SBO     0   NOT=    THE COMPILER USES THIS TABLE
1711   05770  004112        DEF  SBO     1   IMP*
1712   05771  004112        DEF  SBO     2   =<
1713   05772  004112        DEF  SBO     3   =
1714   05773  004112        DEF  SBO     4   >
1715   05774  004112        DEF  SBO     5   *
1716   05775  004112        DEF  SBO     6   /
1717   05776  004114        DEF  PLU     7   +
1718   05777  004116        DEF  MIN     8   -
1719   06000  004115        DEF  IGNR    9   U-
1720   06001  004126        DEF  FCN    10   SQRT
1721   06002  004107        DEF  GAZ    11   ←
1722   06003  004126        DEF  FCN    12   R
1723   06004  004462        DEF  NUM    13   0
1724   06005  004462        DEF  NUM    14   1—9
1725   06006  004462        DEF  NUM    15   .
1726   06007  004462        DEF  NUM    16   EEX
1727   06010  004130        DEF  OPD    17   A B C X Y Z
1728   06011  004136        DEF  LBR    18   (
1729   06012  004155        DEF  RBR    19   )
1730   06013  004157        DEF  COM    20   ,
1731   06014  004426        DEF  QUO    21   ;
1732   06015  004207        DEF  IF     22   JMP
1733   06016  004254        DEF  EOS    23   :
1734   06017  004373        DEF  GTO    24   GSB
1735   06020  004373        DEF  GTO    25   GTO
1736   06021  004241        DEF  EOL    26   EOL
1737   06022  004221        DEF  ENT    27   ENT
1738   06023  004213        DEF  RET    28   RET
1739   06024  004213        DEF  RET    29   STOP
1740   06025  004217        DEF  SFFNT  30   NORMAL
1741   06026  004217        DEF  SFFNT  31   TRACE
1742   06027  004217        DEF  SFFNT  32   FIXED
1743   06030  004217        DEF  SFFNT  33   FLOAT
1744   06031  004166        DEF  STF    34   SET FLAG
1745   06032  004126        DEF  FCN    35   FLAG
1746   06033  004207        DEF  IF     36   IF
1747   06034  004217        DEF  SFFNT  42   END
1748   06035  004215        DEF  PD     38   PRINT
1749   06036  004215        DEF  PD     39   DISPLAY
1750   06037  004211        DEF  LR     40   LOAD
1751   06040  004211        DEF  LR     41   RECORD
1752   06041  004213        DEF  RET    42   END
1753   06042  001362        DEF  EXIT   43   OPTION #1
1754   06043  001362        DEF  EXIT   44   OPTION #2
1755   06044  001362        DEF  EXIT   45   OPTION #3
1756   06045  004115        DEF  IGNR   46   ALPHA
1757   06046  004115        DEF  IGNR   47   NULL
1758   06047  004115        DEF  IGNR   48   IGNORE
1759   06050  004170        DEF  STF+2  49   CLEAR FLAG
1760   06051  004110        DEF  UPARO  50   ↑
```

150

ADDRESS TABLE FOR INTERPRETER ROUTINES

```
1651*
1652  05701  005702 TAB3    DEF *+1
1653*
1654  05702  062615          JSM XBSET    0 NOT =     THE INTERPRETER USES THIS
                                                      TABLE
1655  05703  062615          JSM XBSET    1 IMP*
1656  05704  062615          JSM XBSET    2 =<
1657  05705  062615          JSM XBSET    3 =
1658  05706  062615          JSM XBSET    4 >
1659  05707  062615          JSM XBSET    5 *
1660  05710  062615          JSM XBSET    6 /
1661  05711  062615          JSM XBSET    7 +
1662  05712  062615          JSM XBSET    8 -
1663  05713  062631          JSM XUSET    9 U-
1664  05714  062631          JSM XUSET   10 SQRT
1665  05715  066664          JMP XGAZ    11 ←
1666  05716  066702          JMP SUBSC   12 R
1667  05717  074006 FROT      OCT 074006
1668  05720  105767 TAB1      DEF TTB1,I
1669  05721  106051 TAB2      DEF TTB2-1,I
1670  05722  116353 TAB5      DEF TTB5,I
1671  05723  066647          JMP XOPD    17 A B C X Y Z
1672  05724  001606 ADPS      DEF PS      18 (
1673  05725  005142 ADSP      DEF RDSPY   19 )
1674  05726  066570          JMP XJ      20 ,
1675  05727  066722          JMP XQUO    21 "
1676  05730  067041          JMP XJMP    22 JUMP
1677  05731  066564          JMP XR      23 ;
1678  05732  067210          JMP XGSB    24 GSB
1679  05733  067213          JMP XGTO    25 GTO
1680  05734  067047          JMP XEOL    26 EOL
1681  05735  164534          JMP XENT,I  27 ENT
1682  05736  067152          JMP XRET    28 RET
1683  05737  061250          JSM XSTOP   29 STOP
1684  05740  061256          JSM XNORM   30 NORMAL
1685  05741  061262          JSM XTRAC   31 TRACE
1686  05742  062764          JSM FIX     32 FIXED
1687  05743  062766          JSM FLO     33 FLOAT
1688  05744  067437          JMP STFI    34 SET FLAG
1689  05745  067414          JMP STFF    35 FLAG
1690  05746  067026          JMP XIF     36 IF
1691  05747  063006          JSM SPA     37 SPACE
1692  05750  067071          JMP XPRNT   38 PRINT
1693  05751  067073          JMP XDISP   39 DISPLAY
1694  05752  164225          JMP LOADX,I 40 LOAD
1695  05753  160226          JSM RCRDX,I 41 RECORD
1696  05754  161710          JSM XEND,I  42 END
1697  05755  165634          JMP OPT1,I  43 OPTION #1
1698  05756  165635          JMP OPT2,I  44 OPTION #2
1699  05757  165636          JMP OPT3,I  45 OPTION #3
1700  05760  067537          JMP XNUM    46 ALPHA
1701  05761  066570          JMP XJ      47 NULL
1702  05762  067067          JMP NONX    48 ABORT
1703  05763  067441          JMP CLFI    49 CLEAR FLAG
1704  05764  007355 DRECP     DEF 7355B
1705  05765  000000          NOP         CHECK SUM
1706  05766  066570          JMP XJ      52 END OF ENTER STMT
```

(154)

INTERPRETER SUBROUTINES

```
1762*
1763   06052  004225  TTB2    DEF  GE1     TABLE OF G-ROUTINES FOR COMPILER
1764   06053  004474          DEF  GE2
1765   06054  006072          DEF  GE3
1766   06055  001300          DEF  CL
1767   06056  004450          DEF  GE5
1768   06057  006064          DEF  GE6
1769   06060  001300          DEF  CL
1770   06061  004413          DEF  GE8
1771   06062  004407          DEF  GE9
1772   06063  004172          DEF  GE10
1773*
1774*  G=6 GO TYPE
1775*
1776   06064  014452  GE6     CPB  CNFQ
1777   06065  066101          JMP  GQT
1778   06066  014375          CPB  CNFA
1779   06067  066103          JMP  GRL
1780   06070  014374          CPB  CNFS
1781   06071  066103          JMP  GRL
1782*
1783*  G=3  GO≠BEGIN
1784*
1785   06072  014366  GE3     CPB  CNFD
1786   06073  066077          JMP  SET9
1787   06074  014367          CPB  CNFZ
1788   06075  066077          JMP  SET9
1789   06076  065317          JMP  E1
1790*
1791   06077  024373  SET9    LDB  DP9
1792   06100  066104          JMP  *+4
1793   06101  024375  GQT     LDB  DP7
1794   06102  066104          JMP  *+2
1795   06103  024401  GRL     LDB  DP3
1796   06104  035602          STB  G
1797   06105  065304          JMP  CP1
1798*
1799*
1800*  ADDRESS 6106B THROUGH 6116B
1801*  ARE USED BY INEQUALITIES
1802*
1803   16353                   ORG  16353B
1804*
1805   16353  005274  TTB5    DEF  GABS
1806   16354  005303          DEF  GMNS
1807   16355  005306          DEF  GPLS
```

CONTROL SUPERVISOR SUBROUTINES

```
0254*********************************
0255*   CONTROL SUPERVISOR
0256*
0257*     8/3/70
0258*     EEO
0259*
0260*   THE CONTROL SUPERVISOR (.CS) DIRECTS ALL KEYCODES
0261*   LOADED BY THE KEYBOARD-INTERRUPT ROUTINE TO THE
0262*   PROPER HANDLING ROUTINE.
0263*
0264**********************
0265*
0266    16000                        ORG  16000B
0267*
0268    16000  177744  M28           DEC  -28
0269*
0270    16001  021563  INTRR         LDA  TSW        TSW=4 IF RUNNING USER PROGRAM
0271    16002  006000                ADB  M28        IF INTERRUPT MODE AJUST CN KEYCODE
                                                      EXECUTION
0272    16003  010400                CPA  P4
0273    16004  066016                JMP  PRG+2      YES INTERRUPT MODE,LEAVE AS IS
0274    16005  066014                JMP  PRG        NOT INTERRUPT,SET CN=11 FOR
                                                      PROGRAMMABLE KEY
0275*
0276    16006  160254  SMON          JSM  254B,I
0277    16007  000406  CS            ADA  M2         KEYCLASS IS IN A-REG;
                                                      0=PROGRAMMABLE,1=OPTION
0278*                                                 2=PROGRAM INTERRUPTABLE,
                                                      3=CONTROL
0279    16010  071450                SZA  INTRR      IF A-REG=0,CODE IS PROGRAM
                                                      INTERRUPTABLE
0280    16011  000403                ADA  P1
0281    16012  070110                SZA  PRG        AFTER ADDING +1 A-REG=0 INDICATES
                                                      OPTION BLOCK
0282    16013  070613                SAP  CONTR      A-REG + ,KEYCODE IS CONTROL CLASS
0283*
0284    16014  024371  PRG           LDB  P11        SET CN=11 FOR PROGRAMMABLE KEYS
0285    16015  021563                LDA  TSW        GET TABLE ADDRESS=BASE+(TSW+6(CN))
0286    16016  035717                STB  TEMP1        THE CONTENTS OF THE ADDRESS
0287    16017  074704                SBL  2            POINTS TO THE PROCESSING ROUTINE.
0288    16020  005717                ADB  TEMP1        THIS TABLE OF ADDRESSES CAN BE
0289    16021  005717                ADB  TEMP1        CONSIDERED AS A MATRIX OF SIZE
0290    16022  050365                AND  MASK7        (6,11)WHERE EACH ELLEMENT IS FOUND
0291    16023  070037                ADB  A            BY (TSW,CN) WHERE TSW IS THE
                                                      CONTROLLING FLAG FOR .CS AND CN
                                                      IS THE MODIFIED CONTROL NUMBER
                                                      FOUND IN "MTABL"
0292    16024  006044                ADB  .ATBL      B-REG HAS ADDRESS OF PROCESSING
                                                      ROUTINE
0293    16025  021563                LDA  TSW
0294    16026  074737                JMP  B,I        JUMP TO THE ROUTINE
0295*
0296    16027  004422  CONTR         ADB  M25        CONTROL KEYS (CN) BIASED BY +26
0297    16030  035717                STB  TEMP1      SAVE MODIFIED CONTROL NUMBER
0298    16031  021563                LDA  TSW
0299    16032  010400                CPA  P4           IN PROGRAM RUNNING MODE?(TSW=4?)
0300    16033  170402                RET               YES,DO NOT PROCESS
0301    16034  024354                LDB  UNC1         SET KSTOR SWITCH FOR
0302    16035  035621                STB  KSTOR          REPLACEMENT STORAGE VS INSERT
                                                          STORAGE
```

```
0303  16036  025563          LDB  TSW
0304  16037  014401          CPB  P3      IF INSERT MODE(TSW=3)
0305  16040  024403          LDB  P1        RESET TSW=1
0306  16041  035563          STB  TSW
0307  16042  025717          LDB  TEMP1   RESTORE CN TO B-REG
0308  16043  066017          JMP  PRG+3    GO COMPUTE PROCESS ADDRESS WITH
                                           OLD TSW
0309  16044  106110  .ATBL   DEF  .TB3-7,I
```

ADDRESS TABLE FOR CONTROL SUPERVISOR ROUTINES

```
0311  06117                 ORG 6117B
0312*
0313  06117  006247  .TB3    DEF A.11  TSW=1  (KEYBOARD ENTRY MODE)        EXECUTE KEY (1)
0314  06120  006247          DEF. A.11     2  (RECALL MODE)                 "       "
0315  06121  006247          DEF A.11      3  (INSERT CHARACTER MODE)       "       "
0316  06122  001250          DEF STP       4  (RUN PROGRAM MODE)           "       "
                                              (INTR.)                      "       "
0317  06123  006255          DEF A.51      5  (ENTER MODE)                          STOP KEY
0318  06124  006247          DEF A.11      6  (EOL MODE)                    "       "
0319*                                                                      "       "
0320  06125  006272          DEF A.12      1
0321  06126  006272          DEF A.12      2                                STORE KEY  (2)
0322  06127  006265          DEF A.32      3                                "       "
0323  06130  001256          DEF B.72      4  NORMAL KEY                    "       "
0324  06131  100533          DEF RET1,I    5  IGNORE                       "       "
0325  06132  006272          DEF A.12      6
0326*                                                                      "       "
0327  06133  006461          DEF A.13      1
0328  06134  006463          DEF A.23      2                                INSERT KEY (3)
0329  06135  006460          DEF A.33      3                                "       "
0330  06136  001262          DEF B.73      4  TRACE KEY                     "       "
0331  06137  006463          DEF A.23      5                                "       "
0332  06140  006456          DEF A.63      6
0333*                                                                      "       "
0334  06141  006413          DEF A.14      1
0335  06142  006412          DEF A.24      2                                CLEAR KEY  (4)
0336  06143  006413          DEF A.14      3
0337  06144  100305          DEF FXD,I     4                                "       "
0338  06145  006414          DEF A.54      5
0339  06146  006413          DEF A.14      6                                "       "
0340*                                                                      "       "
0341  06147  006537          DEF A.65      1
0342  06150  006523          DEF A.25      2                                DELETE KEY (5)
0343  06151  006540          DEF A.35      3                                "       "
0344  06152  100304          DEF FLT,I     4                                "       "
0345  06153  006537          DEF A.65      5
0346  06154  006537          DEF A.65      6
0347*                                                                      "       "
0348  06155  006423          DEF A.16      1
0349  06156  006427          DEF OK4       2                                RECALL KEY (6)
0350  06157  006423          DEF A.16      3                                "       "
0351  06160  016373          DEF B.77      4  SFG KEY                       "       "
0352  06161  100533          DEF RET1,I    5  IGNORE                       "       "
0353  06162  006427          DEF OK4       6
0354*                                                                      "       "
```

```
0355  06163  006467          DEF  A.17     1          FORWARD  (7)
0356  06164  006504          DEF  A.27     2          "
0357  06165  006467          DEF  A.17     3          "
0358  06166  007754  .KBR3   DEF  .KBR
0359  06167  006467          DEF  A.17     5          "
0360  06170  006467          DEF  A.17     6          "

0362  06171  006435          DEF  A.18     1          RACKSPACE  (8)
0363  06172  006437          DEF  A.28     2          "
0364  06173  006434          DEF  A.38     3
0365  06174  000000          BSS  1                   CKECKSUM WORD
0366  06175  006437          DEF  A.28     5
0367  06176  006435          DEF  A.18     6
0368*                                                 LIST KEY  (9)
0369  06177  006562          DEF  A.19     1          "  "
0370  06200  006565          DEF  L2       2
0371  06201  100533          DEF  RET1,I   3          IGNORE
0372  06202  016014  .PRG     DEF  PRG
0373  06203  100533          DEF  RET1,I   5          IGNORE
0374  06204  006565          DEF  L2       6          "  "
0375*                                                 RUN PROGRAM KEY (10)
0376  06205  006656          DEF  A.110    1
0377  06206  006655          DEF  A.210    2
0378  06207  100533          DEF  RET1,I   3          IGNORE
0379  06210  020000  B13     OCT  20000
0380  06211  006775          DEF  ASSG     5          "  "
0381  06212  006661          DEF  A.110+3  6
0382*                                                 PROGRAMMABLE  (11)
0383  06213  007066          DEF  P02      1          "
0384  06214  007066          DEF  P02      2          "
0385  06215  007066          DEF  P02      3
0386  06216  006245          DEF  RET      4          "
0387  06217  007061          DEF  A.511    5
0388  06220  007066  TBLE    DEF  P02      6
0389*
0390  16356                  ORG  16356B
0391  16356  101631  TB1     DEF  1631B,I             POINTER TO OPTION BLOCK SYNTAX
                                                      ROUTINE #1
0392  16357  101632          DEF  1632B,I             FOR OPTION BLOCK #2
0393  16360  101633          DEF  1633B,I             FOR OPTION BLOCK #3
```

(152)

SYSTEM MONITOR ROUTINE

```
0395*
0396*******************************
0397* THE FOLLOWING IS THE MAINLINE FOR THE KEYBOARD,I.E.,.NKK
0398*
0399  06221                    ORG TBLE+1
0400*
0401  06221 160251 .NKK        JSM  .DSP,I   CALL DISPLAY BRIVER (ROUTINE IN I/O
                                               SUP)
0402  06222 160323 .CS         JSM  .SMON,I  PROCESS KEYCODE
0403  06223 062225             JSM  KY6      SETUP DISPLAY BUFFER(NORMAL RETURN)
0404  06224 066221 KY5         JMP  .NKK     DISPLAY LINE OR RESULT(ALPHANUMERIC
                                               RESULT RETURN)
0405*
0406  06225 035577 KY6         STB  ERST     B=0 LINE AS COMPILED OKAY:-1 AN ERROR
                                               OCURRED
0407  06226 021705             LDA  .WMOD
0408  06227 070712             SAM  RET      IF NEW KEY RETURN
0409*
0410  06230 070742 CR0         SAR  16        ELSE BUILD DISPLAY
0411  06231 031571             STA  LINCT    ZERO CHARACTER COUNTER OF LINE
0412  06232 063112             JSM  CLDB      CLEAR DISPLAY BUFFER
0413  06233 062242             JSM  CTSW2     TSW=2?
0414  06234 063116 CR2         JSM  .STLN    YES, BUILD LINE # FOR DISPLAY
0415  06235 067210             JMP  .LOC.    BUILD DISPLAY
0416*
0417  06236 021542 .I8         LDA  ENDS     .I8 IS A UTILITY ROUTINE FOR SETTING
                                               SYSTEM STACK
0418  06237 063153             JSM  FM11     ROUTINE STARTS AT ENDS LOOKING FOR
0419  06240 031543             STA  LIM1      NEXT -1 WHICH MARKS BEGINNING OF
                                               MAIN PROGRAM
0420  06241 072230             RIA  RET      + 1 NECESSARY FOR MAG-CARD ROUTINE
                                               AND END
0421*
0422  06242 021563 CTSW2       LDA  TSW
0423  06243 050365             AND  MASK7
0424  06244 010402             CPA  P2
0425  06245 170402 RET         RET
0426  06246 164533             JMP  RET1,I
```

```
0428*
0429*********************************
0430*
0431*  THE FOLLOWING ROUTINE PROCESSES THE EXECUTE KEY.
0432*
0433   06247  063646  A.11      JSM  RKBUF    EXECUTE KEYBOARD BUFFER;IF NO-STOP
                                                 USE SAME POINTER IN PROGRAM AREA
0434   06250  066646            JMP  RU71     (STOP RETURN)
0435   06251  062253            JSM  TSW6     SET TSW=6
0436   06252  164533            JMP  RET1,I
0437*
0438   06253  020376  TSW6      LDA  P6
0439   06254  067576            JMP  EX1      TSW=(1)6
0440*
0441   06255  063647  A.51      JSM  RKBUF+1  EXECUTE KEYBOARD BUFFER (NO INTRP)
0442   06256  066263            JMP  TSW5+2   NORMAL RETURN
0443   06257  062261            JSM  TSW5     ERROR RETURN
0444   06260  164533            JMP  RET1,I
0445*
0446   06261  020377  TSW5      LDA  P5
0477   06262  067576            JMP  EX1      TSW=(1)5
0448*
0449   06263  062261            JSM  TSW5
0450   06264  066647            JMP  .A7
```

```
0452*
0453*******************************
0454*
0455*  THE FOLLOWING ROUTINES PROCESS THE STORE KEY.
0456*
0457   06265  063557  A.32    JSM   EXEQT
0458   06266  074453          SBP   STR2
0459   06267  021563  .I12    LDA   TSW
0460   06270  040262          IOR   BT14
0461   06271  067577          JMP   EX1+1    IF ERROR ON EOL FOR STORE SET BIT14
                                               IN TSW
0462*
0463   06272  063557  A.I2     JSM   EXEQT
0464   06273  075612          SBM   .I12
0465   06274  062242          JSM   CTSW2    TSW=2?
0466   06275  063406          JSM   DRCMA    YES,STORE IN CURRENT LINE
0467   06276  063472          JSM   DELT     DELETE OLD LINE
0468   06277  045567  STR2    ISZ   MPH      SET FOR SLOW GTO'S
0469   06300  074742          SBR   I6
0470   06301  061352          JSM   RBAD
0471   06302  020535          LDA   TOPS     COMPUTE#-OF-WORDS USED TO
0472   06303  070076          TCA              COMPILE THE CURRENT INPUT LINE.
                                               TOPS-PSP=#OF WORDS
0473   06304  001545          ADA   PSP
0474   06305  001542          ADA   END$     INCREASE END-OF-PROGRAM MARKER
                                               THE NECESSARY AMOUNT
0475   06306  031722          STA   TEMP2    SAVE NEW ENDS
0476   06307  031725          STA   TEMP3    TEMP3 USED IN EXPANDING MEMORY
                                               FOR INSERTING NEW LIN
0477   06310  001615          ADA   BSTK     SYSTEM STACK MOVES WITH END$. SEE
0478   06311  070056          CMA              IF NEW LINE FITS IN USER MEMORY.
0479   06312  001377          ADA   MAW        MAW-(#-OF-WORDS+END$+72+1)IF
0480   06313  070413          SAP   .I0-1      POSITIVE LINE FITS ELSE ERROR
0481   06314  024370          LDB   MXERR    ERROR #12 FOR MEMORY OVERFLOW
0482   06315  061351          JSM   BAD      SETUP .WMOD (ROUTINE ON BASE
                                               PAGE)
0483   06316  063135          JSM   DSNO     BUILD LINE NUMBER
0484   06317  062234          JSM   CR2      CALL ROUTINE TO BUILD LINE WITH
                                               LINE NUMBER
0485   06320  020401          LDA   P3       SET TSW=3
0486   06321  063576          JSM   EX1      SET BIT15 ON TSW(SAYS ERROR=MEMORY
                                               OVERFLOW)
0487   06322  164533          JMP   RET1,I     RETURN FOR DISPLAY
0488*
0489   06323  025542          LDB   END$     SET B-REG AS POINTER TO OPEN
                                               MEMORY FOR NEW LINE
0490   06324  015546  .I0     CPB   CMA      IS OPENING COMPLETE?
0491   06325  066333          JMP   .I1      YES,PREPARE TO INSERT NEW LINE
0492   06326  074517          LDA   B,I      GET IT
0493   06327  131725          STA   TEMP3,I  STORE WORD IN NEW LOCATION
0494   06330  004405          ADB   M1
0495   06331  055725          DSZ   TEMP3
0496   06332  066324          JMP   .I0      RECYCLE
0497*
0498   06333  074517  .I1     LDA   B,I
0499   06334  131725          STA   TEMP3,I
0500   06335  021722          LDA   TEMP2    GET NEW END-OF-PROGRAM MARK
0501   06336  031542          STA   END$     STORE IT IN END$ AND COMPUTE NEW
                                               BSTK
0502*
0503   06337  020535          LDA   TOPS     SET A-REG TO FIRST WORD IN
                                               COMPILER OUTPUT AREA
0504   06340  011545  .I2     CPA   PSP      HAS TRANSFER BEEN COMPLETED?
0505   06341  066402          JMP   .I3      YES
0506   06342  070537          LDB   A,I      NO,GET WORD
0507   06343  135546          STB   CMA,I    STORE IT IN USER PROGRAM AREA
```

```
0508  06344  045546          ISZ  CMA
0509  06345  073570          RIA  .I2
0510*
0511  06346  021546  .I5     LDA  CMA     LOOK TO SEE IF IN DEFINE FUNCTION
                                           AREA
0512  06347  070076          TCA          IF TRUE DON'T CHANGE ENDS
0513  06350  001543          ADA  LIM1    ELSE CHANGE AT LABEL .I7
0514  06351  070452          SAM  .I7
0515  06352  121546          LDA  CMA,I   END WAS STORED IN DEFINE FUNCTION
                                           AREA
0516  06353  010405          CPA  M1      LOOK TO SEE IF CMA IS POINTING TO -1
0517  06354  066357          JMP  .I10    YES,DON'T STORE-2 AT CMA.
0518  06355  020406          LDA  M2      STORE-2.CLEAN-UP ROUTINE IS CALLED
                                           BY END LINK
0519  06356  131546          STA  CMA,I     AND REMOVES OLD PROGRAM FROM
                                           -2 TO A -1
0520  06357  161710  .I10    JSM  .END2,I CALL CLEAN-UP VIA END LINK ON
                                           RETURN FROM END LINK
0521  06360  031600          STA  PMA     CMA IS BACK TO MAIN-PROGRAM.
                                           SET PMA BACK ALSO
0522  06361  066413          JMP  A.14    JUMP TO CLEAR ROUTINE
0523  06362  021546  .I7     LDA  CMA     END WAS FOUND IN INPUT AND MAIN-
                                           LINE-PROGRAM
0524  06363  031542          STA  END$    UPDATE END$
0525  06364  020405          LDA  M1
0526  06365  131542          STA  END$,I
0527  06366  063406  .I6     JSM  DRCMA   MOVE CMA TO POINT TO BEGINNING OF
                                           LAST LINE
0528  06367  020402  STR1     LDA  P2
0529  06370  063044          JSM  NWLNE+1
0530  06371  063540          JSM  UNCP1-1
0531*
0532  06372  063604          JSM  SCAN1
0533  06373  021705          LDA  .WMOD
0534  06374  070151          SLA  *+3
0535  06375  063055          JSM  NW1     TRACE MODE PRINT PROGRAM LINE
0536  06376  062616          JSM  L5
0537  06377  020402          LDA  P2
0538  06400  063576          JSM  EX1
0539  06401  164303          JMP  OUT,I
0540*
0541  06402  063055  .I3     JSM  NW1     RESET CP TO BEGINNING OF INPUT
                                           STRING
0542  06403  062236          JSM  .I8     ADJUST LIM1 POINTER(NECESSARY IF
                                           DEFINE FUNCTION)
0543  06404  063457  .I4     JSM  FETCH   LOOK TO SEE IF END WAS IN INPUT STRING
0544  06405  010465          CPA  ENLNE   EOL?
0545  06406  066366          JMP  .I6     YES,SEARCH OF INPUT STRING COMPLETED
0546  06407  010457          CPA  END.    IS CHARACTER END?
0547  06410  066346          JMP  .I5     YES, PREPARE TO UPDATE END$
0548  06411  066404          JMP  .I4     NO,CONTINUE SEARCH OF INPUT STRING
```

```
0550*
0551*******************************
0552*
0553*  THE FOLLOWING ROUTINES PROCESS THE CLEAR KEY.
0554*
0555   06412  063406  A.24    JSM   DRCMA
0556*
0557   06413  020376  A.14    LDA   P6        SET TSW=6
0558*
0559   06414  063044  A.54    JSM   NW0-1     A-REG INTO TSW;CLEAR INPUT AREA
0560   06415  067557          JMP   EXEQT     PUT EOL IN INPUT
0561*
0562   06416  020376  CLR1    LDA   P6        CLEAR CODE FOR OPTION BLOCKS
0563   06417  063020          JSM   SKCD+1    SETUP CODE AND CN
0564   06420  161560          JSM   OSW,I     SEND CLEAR CODE TO PTION BLOCKS
0565   06421  021544          LDA   SKEY
0566   06422  067020          JMP   SKCD+1    RESET CODE AND CN FOR THE PRESENT
                                               KEYCODE
```

```
0568*
0569*  THE FOLLOWING ROUTINE PROCESS THE RECALL KEY.
0570*
0571  06423 063363 A.16     JSM  CKERR    IF SYNTAX ERROR — NOP
0572  06424 063646          JSM  RKBUF    EXECUTE INPUT LINE
0573  06425 066427          JMP  *+2      NORMAL RETURN
0574  06426 170402          RET           ERROR RETURN
0575  06427 063363 OK4      JSM  CKERR
0576  06430 121546          LDA  CMA,I
0577  06431 010405          CPA  M1       ARE WE AT END-OF-PROGRAM
0578  06432 066600          JMP  SPREG    YES,OUTPUT # OF R-REG
0579  06433 066367 RCLL     JMP  STR1
0580*
0581*  THE FOLLOWING ROUTINES PROCESS THE BACKSPACE KEY
0582*
0583*
0584  06434 070742 A.38     SAR  16       ENTRY IF MEMORY OVERFLOW AND INSERT
0585  06435 050262 A.18     AND  252      SAVE BITS 15&14
0586  06436 040403          IOR  P1       PREPARE TO SET TSW=1
0587  06437 070073 A.28     SAP  *+1,C    CLEAR SIGN BIT. ENTRY FOR RECALL MODE
0588  06440 031563          STA  TSW      SET TSW
0589  06441 063355          JSM  DRECP    NORMAL DRECREMENT OF CP FOR
                                           BACKSPACE
0590  06442 015565          CPB  BP       BACKSPACE OVER BEGINNING-OF-LINE?
0591  06443 066445          JMP  B4       YES
0592  06444 067604 B2       JMP  SCAN1    NO,RECOMPILE CORRECTED LINE
0593*
0594  06445 063367 B4       JSM  INCP     AT BEGINNING OF LINE INCREMENT TO
                                           FIRST POSITION
0595  06446 062242          JSM  CTSW2    TSW=2?
0596  06447 066451          JMP  *+2      YES
0597  06450 067604          JMP  SCAN1    NO JUMP TO RESET COMPILER
0598  06451 063406          JSM  DRCMA    RECALL PRECEEDING PROGRAM LINE
0599  06452 063406          JSM  DRCMA      REQUIRES DRECREMENTING CMA TWICE
0600  06453 010405          CPA  M1       BEGINNING OF PROGRAM AREA
0601  06454 067604          JMP  SCAN1    YES,JUMP TO RESET COMPILER
0602  06455 066433          JMP  RCLL     NO,RECALL NEW PROGRAM LINE
0603*
0604**********************************
0605*
0606*  THE FOLLOWING ROUTINES ALLOW INSERTING CHARACTERS IN A
0607*  RECALLED LINE.
0608*
0609  06456 070253 A.63     SAP  A.23     TSW=6 AND INSERT KEY. IF POSITIVE SET
                                           INSERT MODE
0610  06457 066413          JMP  A.14     ELSE ACT LIKE CLEAR KEY
0611*
0612  06460 070312 A.33     SAM  A.23+3
0613*
0614  06461 040401 A.13     IOR  P3       TSW=1 AND INSERT KEY;SET FOR CHARACTER
                                           INSERT MODE
0615  06462 031563          STA  TSW
0616*
0617  06463 063363 A.23     JSM  CKERR
0618  06464 020357          LDA  ISTOR    GET ADDRESS OF INSERT ROUTINE
0619  06465 031621          STA  KSTOR    STORE IN STORE LINK WORD
0620  06466 164303          JMP  OUT,I
```

```
0622*
0623*******************************
0624*
0625*   THE FOLLOWING ROUTINE PROCESSES THE FORWARD KEY
0626*
0627  06467  071752 A.17    SAM *-1       EOL IF -
0628  06470  070744         SAL 1
0629  06471  070252         SAM A.27P     IF BIT14 CHECK FOR STORE LINE
0630  06472  063361         JSM CKERR-2
0631  06473  021566         LDA CP
0632  06474  063367         JSM INCP      NO ERROR
0633  06475  067611         JMP SCAN3     BRING NEXT CODE INTO SYSTEM
0634*
0635  06476  063361 A.27P    JSM CKERR-2
0636  06477  063457         JSM FETCH
0637  06500  063355         JSM DRECP     MOVE POINTER BACK
0638  06501  010465         CPA ENLNE     IS CHARACTER EOL?
0639  06502  066272         JMP A.12      YES STORE CURRENT LINE
0640  06503  066473         JMP A.17+4    NO PROCESS NEW CHARACTER
0641*
0642  06504  071513 A.27     SAP A.27P
0643  06505  121546         LDA CMA,I     TSW NEGATIVE,LOAD FIRST WORD OF NEXT
                                           LINE
0644  06506  010405         CPA M1        IS IT END OR PROGRAM?
0645  06507  164303         JMP OUT,I     YES,CLEAR A&B REG
0646  06510  020402         LDA P2
0647  06511  063044         JSM NWLNE+1
0648  06512  063540         JSM UNCP1-1
0649  06513  063055         JSM NW1
0650  06514  062416 SCMP     JSM CLR1      RESET OPTION BLOCK COMPILATION
0651  06515  160313         JSM RSCMP,I   RESET COMPILER
0652  06516  035575         STB QOTE
0653  06517  035564         STB ENTSW
0654  06520  035577         STB ERST
0655  06521  061352         JSM RBAD
0656  06522  164303         JMP OUT,I
```

```
0658*
0659*
0660*******************************
0661*
0662*
0663*  THE FOLLOWING ROUTINE PROCESSES THE DELETE KEY
0664*
0665*
0666   06523  070653 A.25     SAP  A.35      IF TSW=2 AND B15=0 THEN DELETE LAST
                                               CHAR SHOWN
0667   06524  063406           JSM  DRCMA     ELSE MOVE CMA BACK TO BEGINNING OF
0668   06525  063472 DELN      JSM  DELT       LINE SHOWN IN PREPARATION FOR
                                                 DELETING LINE
0669   06526  121546           LDA  CMA,I     HAS END$ BEEN MOVED OVER
0670   06527  010405           CPA  M1         THE DELETED LINE?
0671   06530  066532           JMP  D1         YES
0672   06531  066433           JMP  RCLL      NO,DISPLAY LINE THAT NOW OCCUPPIES
                                                 THE DELETED LINE
0673   06532  063406 D1        JSM  DRCMA
0674   06533  000403           ADA  P1        IS MEMORY NULL?
0675   06534  073650           RZA  D1-1      IF NONZERO NO
0676   06535  062413           JSM  A.14      YES,MAKE PROGRAM SEGMENT NULL
0677   06536  066277           JMP  STR2
0678*
0679   06537  071152 A.65      SAM  *-13        IF - RECALL INPUT
0680   06540  070113 A.35      SAP  A.15      DELETE KEY IN MEMORY OVERFLOW;TAKE
                                                 NO ACTION
0681   06541  066277           JMP  STR2        CALL INSERT TO REGENERATE ERROR
                                                   MESSAGE
0682   06542  062544 A.15      JSM  DELC
0683   06543  067604           JMP  SCAN1     RECOMPILE CORRECTED LINE
0684*
0685   06544  025566 DELC      LDB  CP
0686   06545  035573           STB  CP1       SET POINTER FOR FETCHING CODES FROM
                                                 INPUT AREA
0687   06546  063355           JSM  DRECP     MOVE STORAGE POINTER ONE POSITION
0688   06547  035722           STB  TEMP2     SAVE NEXT VALUE FOR CP
0689   06550  015565           CPB  BP        DELETTING BEGINNING LINE CHARACTER?
0690   06551  067367           JMP  INCP      YES,MOVE STORAGE POINTER FORWARD
0691   06552  063453           JSM  FECH      NO,GET CHARACTER TO BE MOVED BACK
0692   06553  070137           LDB  A         MSTOR WORKS WITH CHARACTER IN B-REG
0693   06554  070150           SZA  *+3       LAST CHARACTER IF ZERO
0694   06555  063436           JSM  MSTOR     STORE CHARACTER IN NEW POSITION
0695   06556  066552           JMP  *-4       CONTINUE MOVING INPUT CHARACTERS
0696   06557  063436           JSM  MSTOR     STORE LAST CHARACTER
0697   06560  025722           LDB  TEMP2
0698   06561  067373           JMP  INCP1     RESET INPUT POINTER
```

```
0700*
0701*
0702*******************************
0703*
0704*  THE FOLLOWING ROUTINES PROCESS THE LIST KEY.
0705*
0706*
0707   06562  063646 A.19    JSM  RKBUF    IF INPUT STATEMENT EXECUTE IT
0708   06563  066565         JMP  *+2      NORMAL RETURN
0709   06564  170402         RET           ERROR RETURN
0710*
0711   06565  063363 L2      JSM  CKERR
0712   06566  063135         JSM  DSNO     GET STARTING LINE #
0713   06567  121546         LDA  CMA,I
0714   06570  010405         CPA  M1        ARE WE AT END-OF-PROGRAM
0715   06571  066600         JMP  SPREG       YES,PRINT #OF R-REGISTERS
0716   06572  063045         JSM  NW0      CLEAR INPUT AREA
0717   06573  063541         JSM  UNCP1    UNCOMPILE POLISH CODE
0718   06574  063055         JSM  NW1       SET CP TO BEGINNING CHARACTER
0719   06575  062616         JSM  L5        CALL ROUTINE TO PRINT PROGRAM LINE
0720   06576  045547         ISZ  SNO      INCREMENT LINE #
0721   06577  071413         SAP  L2+2     IF A + RECYCLE;IF - STOP
0722*
0723   06600  068112 SPREG   JSM  CLDB     CLEAR DISPLAY BUFFER
0724   06601  063347         JSM  END+1    SET CMA TO BEGINNING OF MAIN PROGRAM
0725   06602  020467         LDA  R         PLACE "R" IN OUTPUT BUFFER
0726   06603  063114         JSM  .STCD
0727   06604  021542         LDA  END$
0728   06605  001615         ADA  BSTK
0729   06606  070056         CMA           (RR-(END$+BSTK+1))/4=#OF R-REG
0730   06607  001712         ADA  RR
0731   06610  070042         SAR  2
0732   06611  063126         JSM  .STL1+3   CONVERT BINARY NUMBER TO ASCII
0733   06612  160250         JSM  .PRT1,I   PRINT #-OF-R-REGISTERS
0734   06613  020347 L7      LDA  EIGLN     SET UP FOR 8-LINE SPACE(80)
0735   06614  160473         JSM  SPAC,I    CALL SPACE ROUTINE
0736   06615  066413         JMP  A.14     JUMP TO CLEAR ROUTINE. DISPLAY EOL
0737*
0738   06616  063116 L5      JSM  .STLN
0739   06617  160250         JSM  .PRT1,I
0740   06620  063112         JSM  CLDB
0741   06621  020400         LDA  P4
0742   06622  063577         JSM  EX1+1
0743   06623  063516         JSM  CKSTP    CHECK FOR STOP KEY
0744   06624  070752         SAM  .SC3+2   A-REG — IF STOP KEY
0745   06625  020364         LDA  P16       SET # OF CHARACTERS PER LINE TO BE 16
0746   06626  031573         STA  COUNT
0747   06627  063257 .SC1    JSM  .FDCH     ROUTINE COUNTS # OF CHARACTERS PER
0748   06630  070450         SZA  .SC3        CODE RETURNS WITH CODE IN A-REG.
                                               IF ZERO END OF LINE
0749   06631  014405         CPB  M1        B-REG INDICATES STATUS OF BUFFER;-1
                                             BUFFER OVERFLOW
0750   06632  066636         JMP  .SC2        THUS JUMP TO .SC2
0751   06633  063367         JSM  INCP     NO OVERFLOW SET POINTER TO NEXT CODE
0752   06634  063234         JSM  TRANS    GET NMEMONIC FOR CODE
0753   06635  066627         JMP  .SC1     CONTINUE LOOP
0754   06636  063376 .SC2    JSM  .QOTE+1  BUFFER OVERFLOW;WAS OVERFLOW
                                             CHARACTER " YES RESET
0755   06637  160250         JSM  .PRT1,I     QOTE. PRINT LINE
0756   06640  066620         JMP  L5+2     GET REST OF LINE
0757   06641  021573 .SC3    LDA  COUNT     ALL CHARACTERS OF LINE—ARE THERE
0758   06642  010364         CPA  P16         CHARACTERS IN THE DISPLAY BUFFER?
                                               IF COUNT 16 NO CHARACTERS
0759   06643  170402         RET           NO MORE CHARACTERS
0760   06644  164250         JMP  .PRT1,I    YES CHARACTERS IN BUFFER;OUTPUT;
                                               THEN GET NEXT LINE
```

```
0762*
0763*
0764*********************************
0765*
0766*  THE FOLLOWING ROUTINE PROCESSES THE RUN-
0767*  PROGRAM KEY
0768*
0769*
0770   06645  062413  RU7     JSM  A.14      CLEAR INPUT AND SET TSW=6
0771   06646  062253  RU71    JSM  TSW6
0772   06647  160310  .A7     JSM  .FMT2,I   CALL FORMAT ROUTINE TO BUILD RESULT
0773   06650  020405          LDA  M1
0774   06651  025550          LDB  .STPT
0775   06652  031550          STA  .STPT     SET FOR DISPLAYING I/O BUFFER
0776   06653  035572          STB  NRSLT     SAVE OLD .STPT
0777   06654  164533          JMP  RET1,I    RETURN TO KY6
0778*
0779   06655  062413  A.210   JSM  A.14
0780   06656  063646  A.110   JSM  RKBUF     EXECUTE KEYBOARD BUFFER
0781   06657  066661          JMP  *+2       (STOP RETURN)
0782   06660  066251          JMP  A.11+2    ERROR RETURN
0783   06661  021546          LDA  CMA
0784   06662  031600          STA  PMA
0785   06663  020500          LDA  APMA
0786   06664  031605          STA  IMA
0787   06665  063656  RU4     JSM  RPBUF     EXECUTE PROGRAM AREA
0788   06666  066645          JMP  RU7       STOP RETURN
0789   06667  066251          JMP  A.11+2     CHECK TSW=2
```

```
0791*
0792  06670 062715 ENTER  JSM  TENTR     ENTER OKAY TO EXECUTE
0793  06671 067037        JMP  P04+5      NO STMT IS FROM KEYBOARD
0794  06672 062725        JSM  ENT5      YES
0795  06673 066705        JMP  ENT4
0796*
0797  06674 160251 ENT3   JSM  .DSP,I
0798  06675 010553        CPA  RUN       DATA DELIMITER IS RUN PROGRAM KEY
0799  06676 066702        JMP  *+4
0800  06677 160323        JSM  .SMON,I
0801  06700 062225        JSM  KY6       DISPLAY INPUT
0802  06701 066674        JMP  ENT3
0803*
0804  06702 025577        LDB  ERST
0805  06703 075452        SBM  ENT3
0806  06704 160323        JSM  .SMON,I   DATA DELIMITER
0807  06705 074113 ENT4   SBP  *+2
0808  06706 067035        JMP  P04+3     ERROR RECALL PROGRAM LINE
0809  06707 071252        SAM  ENT3       IF A-REG NEGATIVE MORE PARAMETERS
0810*
0811  06710 020400 ENDE   LDA  P4        ELSE COMPLETE
0812  06711 031563        STA  TSW
0813  06712 020532        LDA  XB
0814  06713 031565        STA  BP
0815  06714 164326        JMP  RECY,I
0816*
0817  06715 021624 TENTR  LDA  DISEN      IF NONZERO DO NOT ALLOW ENTER MODE
0818  06716 072210        RZA  *+4
0819  06717 021563        LDA  TSW
0820  06720 010400        CPA  P4        ARE WE IN RUN PROGRAM MODE? (TSW=4)
0821  06721 164533        JMP  RET1,I    YES
0822  06722 164302        JMP  E11,I      NO SET ERROR 11
0823*
0824  06723 062715 ENT2   JSM  TENTR
0825  06724 067034        JMP  P04+2     DON'T EXECUTE ENTER STMT (ERROR RETURN)
0826  06725 121605 ENT5   LDA  IMA,I
0827  06726 031600        STA  PMA
0828  06727 020500 ENT1   LDA  APMA
0829  06730 031605        STA  IMA
0830  06731 063112        JSM  CLDB
0831  06732 020377        LDA  P5
0832  06733 063044        JSM  NW0-1     SET TSW=5 AND READY INPUT BUFFER
0833  06734 024532        LDB  XB
0834  06735 035566        STB  CP         SET CP AT BEGINNING OF INPUT AREA
0835*
0836  06736 160552 STOV1  JSM  SXJ,I      USE INTEPPRETER ROUTINE TO FETCH
                                            CODES FROM ENTER
0837  06737 063375        JSM  .QOTE      SETUP TEST FOR QUOTE FIELD
0838  06740 074110        SZB  *+2         B-REG=1 IF IN QUOTE FIELD
0839  06741 066762        JMP  STOV3       IF PRESENT CHARACTER QUOTE GOTO
                                              STOV3
0840  06742 010507        CPA  ETA       IS CHARACTER END-OF-ENTER STMT MARK?
0841  06743 066767        JMP  ENDV      LAST PARAMETER
0842  06744 010403        CPA  P1        END-OF-ENTER STRING
0843  06745 170402        RET            YES
0844  06746 010545        CPA  COMMA     IS CHARACTER ,?
0845  06747 066767        JMP  ENDV       YES, END OF VARIABLE SETUP FOR DATA
0846  06750 010546        CPA  QUOTE     IS IT "?
```

```
0847  06751  066757          JMP  STOV4     CLOSING QUOTE IGNORE
0848  06752  025544          LDB  SKEY
0849  06753  063436          JSM  MSTOR     SAVE VARIABLE
0850  06754  025564          LDB  ENTSW       WAS THERE A LABEL FIELD?
0851  06755  074351          SLB  STOV2   NO, PLACE VARIABLE IN DISPLAY
0852  06756  066736          JMP  STOV1     YES, GET NEXT CHARACTER
0853  06757  160552  STOV4   JSM  SXJ,I      COMMA IS NEXT CHARACTER IGNORE IT
0854  06760  045564          ISZ  ENTSW     SET LABEL FIELD FLAG
0855  06761  066736          JMP  STOV1
0856  06762  010546  STOV3   CPA  QUOTE      IF CODE IS QUOTE IGNORE
0857  06763  066736          JMP  STOV1     GET NEXT CODE
0858  06764  021544  STOV2   LDA  SKEY    PLACE CHARACTER IN A-REG
0859  06765  063234          JSM  TRANS   GET NMEMONIC AND PLACE IN DISPLAY
                                          BUFFER
0860  06766  066736          JMP  STOV1   RECYCLE
0861*
0862  06767  024475  ENDV    LDB  ENSTM   VARIABLE HAS BEEN FOUND
0863  06770  063436          JSM  MSTOR     STORE: AFTER VARIABLE
0864  06771  075712          SBM  *-2     MAKE CP POINT TO LEFT HALF-OF-WORD
0865  06772  035565          STB  BP      STORE POINTER IN BP THIS PROTECTS
                                          SAVING OF VARIABLE
0866  06773  063055          JSM  NW1     STORE BEGINNING-OF-LINE CHARACTER
                                          AT BP
0867  06774  067566          JMP  EX3     EXECUTE AN EOL AND RETURN FOR DATA
                                          ENTRY
0868*
0869  06775  070253  ASSG    SAP  *+5     CONTINUE ENTRY. IF A + STORE DATA IN
                                          VARIABLE
0870  06776  021430          LDA  FLG       ELSE SET FLAG 13
0871  06777  042210          IOR  B13
0872  07000  031430          STA  FLG
0873  07001  066727          JMP  ENT1    GET NEXT PARAMETER IN ENTER STMT
0874  07002  020532          LDA  XB
0875  07003  031573          STA  CP1     SET POINTER TO RETRIEVE VARIABLE
0876  07004  024464          LDB  ASSM      GAZINTA FOR END OF KEYBOARD ENTRY
0877  07005  063436  ASSG3   JSM  MSTOR     STORE CODES AT END OF INPUT STRING
0878  07006  063453          JSM  FECH      FETCH VARIABLE CODES
0879  07007  070137          LDB  A
0880  07010  010475          CPA  ENSTM     IS IT ;?
0881  07011  067013          JMP  ASSG2     YES, ALL VARIABLES RETRIEVED
0882  07012  067005          JMP  ASSG3     NO, RECYCLE
0883*
0884  07013  063436  ASSG2   JSM  MSTOR   STORE ";" NEEDED TO COMPILE LINE
                                          PROPERLY
0885  07014  063647          JSM  RKBUF+1 EXECUTE INPUT
0886  07015  066727          JMP  ENT1    NORMAL RETURN
0887  07016  065357          JMP  1357B     ERROR RETURN
```

```
0889*
0890  07017  031544 SKCD   STA  SKEY     THIS ROUTINE IS USED TO SETUP SKEY,
                                           CODE, AND CN
0891  07020  000377        ADA  MTABL      (CONTROL ≠). ENTER WITH ASCII CODE IN
                                           A-REG
0892  07021  070517        LDA  A,I        ADD TABLE ADDRESS
0893  07022  031561        STA  CODE       GET CODE WORD.16 BIT WORD DIRECTS
                                           KEYCODE THRU SYSTEM
0894  07023  025561 SKCD1  LDB  CODE     RE-ENTERED HERE AFTER GOING TO
                                           OPTION BLOCKS
0895  07024  021561        LDA  CODE
0896  07025  070342        SAR  8
0897  07026  050401        AND  MASK1      POSITION WORD FOR KEYCODE CLASS
                                         CLASS IN A-REG, 0=PROGRAM, 1=OPTION,
                                           2=INTER., 3=CONTROL
0898  07027  074442        SBR  10         POSITION WORD FOR CONTROL ≠,CN
0899  07030  035540        STB  CN       STORE IN CN
0900  07031  170402        RET
0901*
0902  07032  062514 PO4    JSM  SCMP
0903  07033  061250        JSM  STP
0904  07034  055777        DSZ  RSTAK      ERASE RETURN OF CALLING ROUTINE FOR
                                           ENTER
0905  07035  020500        LDA  APMA
0906  07036  031605        STA  IMA
0907  07037  160552        JSM  SXJ,I      GET CODE FROM PROGRAM AREA
0908  07040  010457        CPA  P26        IS IT EOL?
0909  07041  066712        JMP  ENDE+2     YES RETURN TO INTERPRETER
0910  07042  067037 PO5    JMP  *-3      RECYCLE
```

```
0912*
0913*****************************
0914*
0915*  THE FOLLOWING ROUTINES PROCESS ALL THE PROGRAMMABLE
0916*  KEYS AND OPTION BLOCK KEYS.
0917*
0918   16347                    ORG 16347B
0919*
0920   16347  004322 OPTN       ADB TAB1          CALL SYNTAX ENTRY ON OPTION BLOCKS
0921   16350  074717            JSM B,I           ALLOWS OPTION BLOCKS TO CHANGE
0922   16351  166352            JMP SKCD2,I        CODE AND CN FOR PROC.

0923   16352  007023 SKCD2      DEF SKCD1
0924*
0925   07043                    ORG PO5+1
0926*
0927   07043  020403 NWLNE      LDA P1            ROUTINE CALLED TO START A NEW LINE
0928   07044  031563            STA TSW           SET TSW=1
0929   07045  062514 NW0        JSM SCMP          RESET COMPILATION; B=0 ON RETURN
0930   07046  025565 NW3        LDB BP            GET ADDRESS OF FIRST WORD IN INPUT
                                                    STRING
0931   07047  070742            SAR 16
0932   07050  015545            CPB PSP           ARE WE UP TO COMPILER OUTPUT AREA?
0933   07051  067055            JMP *+4           YES, COMPLETED ZEROING INPUT AREA
0934   07052  074557            STA B,I           NO, ZERO WORD IN INPUT AREA
0935   07053  004405            ADB M1            MOVE POINTER
0936   07054  067050            JMP *-4           RECYCLE
0937   07055  021565 NW1        LDA BP
0938   07056  031566            STA CP
0939   07057  024466            LDB BGLNE         PLACE BEGINNING CHARACTER IN INPUT
                                                    STRING
0940   07060  067437            JMP MSTOR+1       STORE IT
0941*
0942   07061  025544 A.511       LDB SKEY          ENTRY FOR USING KEYBOARD IN ENTER
                                                    MODE
0943   07062  014537            CPB STOP          IS CODE STOP, USED TO ABORT ENTER
                                                    MODE
0944   07063  067032            JMP PO4           YES
0945   07064  070233            SAP PO2+2,C       NO, IF BIT15=0 NOT FIRDT CHARACTER SO
                                                    CONTINUE
0946   07065  063044            JSM NWLNE+1       BIT15=1 RESET FOR NEW LINE. TSW MUST
                                                    REMAIN 5
0947*
0948   07066  070113 PO2        SAP *+2           NORMAL ENTRY FOR PROCESSING
                                                    PROGRAMMABLE KEYS
0949   07067  063043            JSM NWLNE         BIT15=1 RESET FOR NEW LINE
0950   07070  063363            JSM CKERR
0951   07071  063375            JSM .QOTE         IS CHARACTER PART OF QUOTE FIELD?
0952   07072  076310            RZB PO1             IF YES B=1 ALLOW ANY CODE
0953   07073  063017            JSM SKCD            IF NO B=0. TEST FOR LEGAL CODE
0954   07074  010403            CPA P1            RETURN WITH CLASS IN A-REG. IS CODE
                                                    OPTION BLOCK (A=1)
0955   07075  160551            JSM OPTN1,I       YES CALL OPTION BLOCK SYNTAX
0956   07076  014355            CPB BYPS          NO, CONTROL NUMBER=48?
0957   07077  067101            JMP *+2           YES, OSW SWITCH BYPASS
0958   07100  161560 PO1        JSM OSW,I         SEND CODE TO COMPILER (CALL ROUTINE
                                                    THRU SWITCH OSW)
0959   07101  035722            STB TEMP2         SAVE B-REG (IF NEGATIVE -1 COMPILER
                                                    REJECTED CODE)
0960   07102  021544            LDA SKEY          GET PRESENT KEYCODE
0961   07103  050345            AND MASK9         OPTION BLOCK USES BITS IN SKEY TO
                                                    FLAG COMPILER
0962   07104  070137            LDB A
0963   07105  014437            CPB EXEC          IS CODE EOL?
0964   07106  024465            LDB ENLNE           YES CHANGE IT TO CODE FOR EOL IN
                                                      INPUT STRING
```

(172)

```
0965  07107  161621          JSM  KSTOR,I    STORE CODE IN INPUT AREA. SWITCH
                                               WORD USE FOR REPLACE
0966  07110  025722  PO3     LDB  TEMP2      OR INSERT. RESTORE STATUS OF ERROR
                                               TO B-REG
0967  07111  170402  LAST    RET
```

```
0969*
0970*******************************
0971*
0972*  THE FOLLOWING ROUTINE PROCESSES THE STOP KEY.
0973*
0974   01250                  ORG 1250B
0975*
0976   01250  021705 STP      LDA .WMOD   EXECUTION ROUTINE FOR STOP KEY. GET
                                            WORD WHERE STATUS
0977   01251  040400          IOR P4        IS KEPT. SET STOP BIT (BIT 2)
0978   01252  065260          JMP N1      GOTO STORE WORD
0979*
0980   01253  021705 NSTP     LDA .WMOD   ROUTINE USED TO CLEAR STOP BIT
0981   01254  050411          AND MASK5
0982   01255  065260 BB       JMP N1
0983*
0984*******************************
0985*
0986*  THE FOLLOWING ROUTINE PROCESSES THE NORMAL KEY.
0987*
0988*
0989   01256  021705 B.72     LDA .WMOD
0990   01257  050406          AND M2      CLEAR NORM/TRACE BIT
0991   01260  031705 N1       STA .WMOD
0992   01261  170402          RET
0993*
0994*******************************
0995*
0996*  THE FOLLOWING ROUTINE PROCESSES THE TRACE KEY.
0997*
0998*
0999   01262  021705 B.73     LDA .WMOD
1000   01263  040403          IOR P1      SET TRACE BIT
1001   01264  065260          JMP N1
1002*
```

```
1004*
1005*******************************
1006*
1007   THE FOLLOWING ROUTINES PROCESS THE FIXED ANF FLOAT KEYS
1008*
1009   16367                    ORG  16367B
1010*
1011   16367 154360 .TB4        DEC -10000      TABLE OF CONSTANTS USED IN
1012   16370 176030             DEC -1000        CONVERTING BINARY
1013   16371 177634             DEC -100         ASCII
1014   16372 177766 DM10        DEC -10
1015*
1016*******************************
1017*
1018*
1019*  THE FOLLOWING ROUTINE PROCESSES THE SFG
1020*
1021*
1022   16373 021430 B.77        LDA  FLG        GET FLAG WORD
1023   16374 072051             SLA  *+1,S        SET FLAG 0, INTERRUPT FOR SFG
1024   16375 031430 BBB1        STA  FLG        RETURN TO STOREAGE LOCATION
1025   16376 170402             RET
1026*
1027   16377 000000             BSS  1          CHECKSUM WORD
1028*                                          SYSTEM MONITOR ROUTINE (CONT.
1029   07112                    ORG  LAST+1
1030   07112 020207 CLDB        LDA  .DBUF      LOAD ADDRESS OF DISPLAY BUFFER
1031   07113 164255             JMP  LI8A,I     CALL ROUTINE TO CLEAR THIS BUFFER
1032*
1033   07144 031720 .STCD       STA  K          A-REG HAS CODE TO BE STORED IN
                                                   DISPLAY BUFFER
1034   07115 164252             JMP  .BF3D,I    CALL ROUTINE TO STORE THIS
                                                   CHARACTER IN DISPLAY BUFR
```

(17A)

```
1036*
1037*  THE FOLLOWING ROUTINES ARE THE DISPLAY AND
1038*  PRINTER ROUTINES USED TO OUTPUT USER SYSTEM COMMANDS.
1039*
1040  07116  063112  .STLN  JSM  CLDB   ROUTINE USED TO BUILD LINE #
1041  07117  063123         JSM  .STL1   CALL ROUTINE TO CONVERT BINARY #
                                          TO ASCII
1042  07120  063204         JSM  COLN    PLACE COLON WITH #
1043  07121  067114         JMP  .STCD   STORE : IN DISPLAY
1044*
1045  07122  063135  STLN   JSM  DSNO
1046*
1047  07123  070742  .STL1  SAR  16       ZERO LOCATION WHICH WILL HOLD #
1048  07124  031571         STA  LINCT     OF CHARACTERS IN LINE #
1049  07125  021547         LDA  SNO     GET CURRENT LINE #
1050  01726  063160         JSM  SBA       CONVERT IT
1051  07127  063114         JSM  .STCD   STORE AN ASC11 CHARACTER
1052  07130  021725         LDA  TEMP3   GET POINTER TO TABLE OF DECIMAL
                                          CONSTANTS
1053  07131  010460         CPA  ENDSR     ARE FINISHED WITH CONVERSION
                                            PROCESS?
1054  07132  170402         RET            YES
1055  07133  063165         JSM  CBA       NO,CONTINUE CONVERSION PROCESS
1056  07134  067127         JMP  *-5     RECYCLE
1057*
1058  07135  070742  DSNO   SAR  16       ROUTINE USED TO DETERMINE CURRENT
1059  07136  031547         STA  SNO       LINE #.CLEAR LINE #.
1060  07137  021546         LDA  CMA     ADDRESS OF CURRENT PROGRAM LINE
1061  07140  063153         JSM  FM11    CALL ROUTINE TO FIND BEGINNING OF
                                          PROGRAM
1062  07141  070070         SIA  *+1     ADDRESS OF FIRST LINE IS A-REG+1 ON
                                          RETURN
1063  07142  011546  DS1    CPA  CMA     IS THAT THE CURRENT LINE?
1064  07143  170402         RET            YES RET
1065  07144  070070  DS2    SIA  *+1     INCREMENT SEARCH POINTER
1066  07145  070537         LDB  A,I     GET NEXT WORD
1067  07146  075713         SBP  DS2     IF BIT15=1 THEN BEGINNING-OF-NEW-LINE
                                          ELSE RECYCLE
1068  07147  045547         ISZ  SNO     FOUND NEW LINE INCREMENT LINE #
                                          COUNTER
1069  07150  014405         CPB  M1      ARE WE AT END-OF-PROGRAM?
1070  07151  170402         RET            YES RETURN TO CHECK IF FOUND
                                            CORRECT LINE
1071  07152  067142  DS3    JMP  DS1
1072*
1073  07153  000405  FM11   ADA  M1      ROUTINE USED TO FIND BEGINNING OF
                                          CURRENT PROGRAM
1074  07154  070537         LDB  A,I       STARTING ADDRESS IS IN A-REG. GET
                                            WORD
1075  07155  014405         CPB  M1      IS IT -1?(-1 IS USED AS BOUNDARY
1076  07156  170402         RET            MARKERS FOR PROGRAM AREA).
                                            YES-FOUND-IT
1077  07157  067153         JMP  *-4     RECYCLE
```

```
1079*
1080   07160  031573  SBA     STA  COUNT   BINARY-ASCII CONVERT SET UP ROUTINE.
                                            #IN A-REG
1081   07161  020474          LDA  CNTAB   ADDRESS OF POWERS OF 10 CONSTANTS
1082   07162  031725          STA  TEMP3   STORE IN POINTER TO THIS TABLE
1083   07163  070742          SAR  16      A-REG ZERO. USED TO SUPPRESS
                                            LEADING ZEROS
1084   07164  067172          JMP  SUBT1-1 GOTO GET FIRST CHARACTER
1085*
1086   07165  020355  CBA     LDA  060     AFTER FINDING FIRST NONZERO
                                            CHARACTER CONT. ENTRY PT
1087   07166  045725          ISZ  TEMP3   INCREMENT POWERS OF TEN TABLE
                                            POINTER
1088   07167  025725          LDB  TEMP3
1089   07170  014460          CPB  ENDSR   ARE WE COMPUTING LAST DIGIT?
1090   07171  067206          JMP  LDIGT     YES LAST DIGIT
1091   07172  025573          LDB  COUNT     NO,GET REMAINDER OF BINARY #
1092   07173  105725  SUBT1   ADB  TEMP3,I SUBTRACT OUT NUMBER OF CURRENT
                                            TENS POSITION DIGIT
1093   07174  074212          SBM  OVFLW     IF MINUS OVERFLOW.TERMINATE DIVIDE
1094   07175  035573          STB  COUNT     NO OVERFLOW SAVE DIGIT
1095   07176  000403          ADA  P1
1096   07177  067173          JMP  SUBT1
1097   07200  071310  OVFLW   SZA  CBA+1   OVERFLOW OCCURRED.IF ZERO NO
                                            NONZERO DIGIT FOUND
1098   07201  040355          IOR  060     MAKE DIGIT ASCII
1099   07202  045571          ISZ  LINCT   INCREMENT CHARACTER COUNTER
1100   07203  170402          RET          RETURN WITH ASCII DIGIT
1101   07204  020470  COLN    LDA  COLLN   SET CHARACTER TO BE COLON
1102   07205  067201          JMP  OVFLW+1
1103   07206  021573  LDIGT   LDA  COUNT   LAST DIGIT, THUS COUNT CONTRAINS
                                            BCD DIGIT
1104   07207  067201          JMP  OVFLW+1 MAKE IT ASCII
1105*
1106   07210  025566  .LOC.   LDB  CP      ROUTINE USED TO DETERMINE
                                            CHARACTERS FOR DISPLAY.
1107   07211  035574          STB  CPE     SAVE CURRENT INPUT STRING
                                            POSITION(IS NEXT AVIALB)
1108   07212  021571          LDA  LINCT   DETERMINE # OF POSITION LINE # WILL
                                            TAKE UP
1109   07213  070076          TCA          16-LINCT
1110   07214  000364          ADA  P16     IS THE # OF POSITIONS LEFT FOR
                                            CHARACTERS IN DISPL
1111   07215  031573          STA  COUNT   STORE RESULT IN COUNT
1112   07216  063355  K2      JSM  DRECP   MOVE CP BACK TO GET KEYCODE FROM
                                            INPUT AREA
1113   07217  063257          JSM  .FDCH   CALL ROUTINE THAT RETRNS WITH
                                            STATUS OF DISPLAY BUFR
1114   07220  010465          CPA  ENLNE   A-REG HAS CURRENT KEYCODE B-REG
                                            STATUS OF BUFR.
1115   07221  067216          JMP  K2      EOL CONTINUE SCAN INPUT STRING
                                            BACKWARD
1116   07222  010466          CPA  BGLNE   BOL CHARACTER?
1117   07223  067230          JMP  STO1      YES START OF INPUTSTRING FOR
1118   07224  075513          SBP  K2        TRANSLATION HAS BEEN FOUND.
1119   07225  014407          CPB  M3        IF B=-3 PRESENT KEYCODE WILL FILL
                                            BUFFER
1120   07226  067230          JMP  STO1      YES,BEGINNING POINTER OF
                                            TRANSLATION PROCESS
1121*
1122   07227  063367  K3      JSM  INCP    B-REG=-1. THUS LAST CHARACTER
                                            WILL OVERFLOW BUFFER
```

(116)

```
1124*
1125  07230  063634  STO1   JSM  .S4       SET INPUT STRING INTO DISPLAY BUFFER
1126  07231  063376         JSM  .QOTE+1     SETUP QUOTE FIELD
1127  07232  063234         JSM  TRANS      TRANSLATE CODE
1128  07233  067230         JMP  STO1       RECYCLE
1129*
1130  07234  063242  TRANS  JSM  .SRB.      CALL ROUTINE WHICH RETURNS WITH
1131  07235  160252         JSM  .BF3D,I      FIRST CHARACTER.STORE IT IN
                                              DISPLAY BUFFER
1132  07236  021722         LDA  M
1133  07237  070610         SZA  .SRB3+2    IF M≠0 MORE CHARACTERS IN MNEMONIC
1134  07240  161620         JSM  NCODE,I    LINK WORD FOR GETTING NEXT
                                              CHARACTERS
1135  07241  067235         JMP  *-4        RECYCLE
1136**********************************
1137  07242  025575  .SRB.  LBD  QOTE      LOAD CONDITION OF QOTE
1138  07243  031720         STA  K          STORE KEYCODE IN K
1139  07244  010465         CPA  ENLNE      IS CODE EOL?
1140  07245  067251         JMP  .SRB3        YES TRANSLATION NECESSARY
1141  07246  010466         CPA  BGLNE      IS CODE ROL?
1142  07247  067251         JMP  .SRB3        YES TRANSLATION NECESSARY
1143  07250  074210         SZB  *+4        IF B-REG=0 NO QUOTE FIELD THUS
                                              TRANSLATE
1144  07251  074742  .SBR3  SBR  16           ELSE NO TRANSLATION
1145  07252  035722         STB  M          SET M FOR ONE CHARACTER
1146  07253  170402         RET
1147  07254  160253         JSM  .BF5A,I    CALL ROUTINE TO GET FIRST CHARACTER
1148  07255  071713         SAP  *-2          IF BIT15 SET ON A-REG THEN
                                                GENERAL MNEMONIC ROUTIN
1149  07256  074737         JMP  B,I          BY OPTION BLOCK.LINK FOR FIRST
                                                CHARACTER B-REG
```

```
1151*
1152   07257  063457 .FDCH   JSM  FETCH    ROUTINE FOR DETERMINE THE #OF
1153   07260  031544         STA  SKEY      KEYCODES THAT WILL FIT IN DISPLAY.
1154   07261  063355         JSM  DRECP       FETCH ADVANCE CP. THUS MOVE IT BACK
1155   07262  070650         SZA  *+13    IF KEYCODE IS ZERO.FIX TO INDICATE NO
                                            CHANGE
1156   07263  010465         CPA  ENLNE   IF EOL OR BOL
1157   07264  067271         JMP  .F1       SPECIAL ACTION IS TAKEN
1158   07265  010466         CPA  BGLNE
1159   07266  067271         JMP  .F1
1160   07267  063375         JSM  .QOTE   GET INFORMATION ABOUT QUOTE FIELD
1161   07270  074150         SZB  *+3       B=0 NO QUOTE FIELD.FIND OUT ABOUT
                                              MULTICHARACTER.
1162   07271  070742 .F1     SAR  16      SPECIAL HANDLING OF EOL
1163   07272  067274         JMP  *+2       AND BOL
1164*
1165   07273  160306         JSM  .BF5G,I CALL ROUTINE FOR DETERMINING # OF
1166   07274  070073         SAP  *+1,C     CHARACTERS FOR THIS CODE.REMOVE
                                              GIT15 IF SET.
1167   07275  000403         ADA  P1      A-REG HAS # OF CHARACTER-1
1168   07276  070076 TCA
1169   07277  001573         ADA  COUNT   SUBTRACT #OF CHARACTER FROM COUNT
                                            (POSITION LEFT)
1170   07300  070412         SAM  .F2       IF MINUS BUFFER OVERFLOW
1171   07301  070250         SZA  .F3       IF ZERO BUFFER JUST FULL
1172   07302  074742         SBR  16      SOME POSITION LEFT THUS B-REG=0 ON
                                            RETURN
1173   07303  031573         STA  COUNT   UPDATE COUNT
1174   07304  021544 .F4     LDA  SKEY    MORE ROOM IN DISPLAY BUFFER
1175   07305  170402         RET
1176*
1177   07306  024407 .F3     LDB  M3        DISPLAY FULL
1178   07307  067303         JMP  .F4-1
1179*
1180   07310  024405 .F2     LDB  M1        LAST CHARACTER OVERFLOWED
1181   07311  067304         JMP  .F4       DISPLAY BUFFER
```

178

```
1183*
1184*  THE FOLLOWING ROUTINES ARE SYSTEM ROUTINES
1185*  FOR THE CONTROL SUPERVISIOR.
1186*
1187*******************************
1188*
1189*
1190  07312  024405 .CSIN   LDB  M1       .CSIN CONTINUATION
1191  07313  135543         STB  LIMI,I      OF THE INITIALIZATION ROUTINE ON
                                              PWR UP.-1 FOR
1192  07314  035431         STB  1431B     SET BOUNDARY FOR COMPILER OUTPUT
                                              AREA
1193  07315  021543         LDA  LIMI        MAIN-PROGRAM BOUNDARY MARKER.
1194  07316  031715         STA  DFUN      INITIALIZE DEFINABLE FUNCTION AREA
                                              TO BE MAIN-PRG.
1195  07317  070070         SIA  *+1       A-REG POINTS TO FIRST WORD OF USER-
                                              PROGRAM AREA
1196  07320  031542         STA  END$      SETUP END$
1197  07321  045542         ISZ  END$        FIRST WORD IN PROG AREA SET TO END
                                              EOL THUS +1 END$
1198  07322  135542         STB  END$,I    SET -1 FOR END-OF-PROGRAM MARKER
1199  07323  024521         LDB  STEN
1200  07324  070577         STB  A,I       STORE "END EOL" FIRST PROGRAM LINE
1201  07325  031600         STA  PMA       SET PMA (INTERPRETER POINTER TO
                                              MAIN-PROG.)LINE 0
1202  07326  021377         LDA  MAW
1203  07327  031712         STA  RR        SET BEGINNING R-REG POINTER TO
                                              HIGHEST ADDRESS
1204  07330  022166         LDA  .KBR3
1205  07331  031711         STA  ILINK     SET INTERRUPT LINKAGE TO NORMAL
                                              KEYBOARD RETURN
1206  07332  020532         LDA  XB
1207  07333  031565         STA  BP        SET INPUT STRING AREA TO BEGINNING
                                              WORD
1208  07334  020354         LDA  UNC1      SET UNCOMPILER OUTPUT TO INPUT
                                              STRING AREA
1209  07335  031552         STA  UNCOT
1210  07336  070742         SAR  16
1211  07337  031625         STA  1625B     CLEAR SECURE PROGRAM PROTECT FLAG
1212  07340  161710         JSM  .END2,I   EXECUTE "END"
1213  07341  061253         JSM  NSTP      CLEAR STOP FLAG
1214  07342  063055         JSM  NW1       NEEDED TO SETUP CP
1215  07343  062433         JSM  RCLL      RECALL LINE 0 OF PROGRAM AREA
1216  07344  063406         JSM  DRCMA     PUT PROGRAM LINE BACK TO FIRST LINE
1217  07345  066223         JMP  .CS+1     JMP CONTROL SUPERVISOR
1218*
1219  07346  061250 END    JSM  STP       ROUTINE IS ONE OF THE PIECES IN THE
                                              TOTAL END LINK
1220  07347  063712         JSM  RGTO
1221  07350  031546         STA  CMA
1222  07351  031604         STA  NMA
1223  07352  074742         SBR  16
1224  07353  035430         STB  FLG       CLEAR ALL "FLAGS"
1225  07354  170402         RET
```

```
1227*
1228*********************************
1229*
1230  07355 025566 DRECP  LDB CP          UTILITY ROUTINE FOR
1231  07356 074672        SBM INCP1,C       DRECEMENT CP
1232  07357 004403        ADB P1
1233  07360 076553        SBP INCP1,S
1234*
1235*********************************
1236*
1237  07361 074742        SBR 16
1238  07362 061352        JSM RBAD      RESET ERROR NUMBER IN .WMOD
1239  07363 025577 CKERR  LDB ERST
1240  07364 074413        SBP INCP1+1   IF ERROR B-REG=-1;IF + RETURN
1241  07365 055777 RET2   DSZ RSTAK
1242  07366 170402        RET
1243*
1244*********************************
1245*
1246  07367 025566 INCP   LDB CP          UTILITY ROUTINE FOR
1247  07370 076153        SBP *+3,S         INCREMENT CP
1248  07371 074073        SBP *+1,C
1249  07372 004405        ADB M1
1250  07373 035566 INCP1  STB CP
1251  07374 170402        RET
1252*
1253*********************************
1254*
1255  07375 021544 .QOTE  LDA SKEY      ROUTINE SETS FLAG FOR PRESENTS FOR
                                          QUOTE FIELD
1256  07376 025575        LDB QOTE      PRESENT KEYCODE IN A-REG.FLAG
                                          (QOTE) IN B-REG
1257  07377 010546        CPA QUOTE
1258  07400 067402        JMP *+2
1259  07401 170402        RET
1260  07402 076111        SLB *+2,S
1261  07403 074071        SLB *+1,C     COMPLEMENTS LSB
1262  07404 035575        STB QOTE        OF B-REG IF A-REG HAS QUOTE KEYCODE
1263  07405 170402        RET
1264*
1265*********************************
1266*
1267  07406 055546 DRCMA  DSZ CMA       UTILITY ROUTINE USED
1268  07407 021542        LDA ENDS
1269  07410 011546        CPA CMA
1270  07411 170402        RET           IF CMA IS POINTING AT PROGRAM END
1271  07412 121546        LDA CMA,I       RETURN TO POINT PROGRAM-LINE-
1272  07413 071553        SAP *-5           POINTER(CMA)TO PRECEEDING LINE
1273  07414 010405        CPA M1
1274  07415 045546        ISZ CMA       IF BOUNDARY MARKER FOUND CORRECT
                                          CMA
1275  07416 170402        RET
```

```
1277*
1278*******************************
1279*
1280   07417  035544  INSTK   STB  SKEY     UTILITY ROUTINE FOR INSERTING
                                              CHARACTERS INTO INPUT
1281   07420  021566          LDA  CP       STRING
1282   07421  031574          STA  CPE      SAVE PRESENT ADDRESS FOR NEW
                                              KEYCODE
1283   07422  063457          JSM  FETCH    MOVE CP TO END OF INPUT STRING
1284   07423  073750          RZA  *-1
1285   07424  063355          JSM  DRECP    MOVE CP BACK TO LAST CHARACTER
1286   07425  063355          JSM  DRECP
1287   07426  063457          JSM  FETCH    GET CHARACTER
1288   07427  070137          LDB  A
1289   07430  063436          JSM  MSTOR     STORE IT ONE POSITION FORWARD
1290   07431  063355          JSM  DRECP    PARTIAL ADJUSTMENT ON CP
1291   07432  015574          CPB  CPE       IS ROOM MAKING COMPLETE?
1292   07433  067435          JMP  *+2      YES, STORE NEW CHARACTER
1293   07434  067424          JMP  *-8
1294   07435  025544          LDB  SKEY
1295*
1296*******************************
1297*
1298   07436  021566  MSTOR   LDA  CP       UTILITY ROUTINE USED TO REPLACE-
1299   07437  070133          SAP  *+2,C      STORE KEYCODES IN THE INPUT
1300   07440  076052          SBM  *+1,C      BUFFER AREA.GET POINTER IN
                                              A-REG.BIT 15 USED TO INDICATE
                                              WHICH HALF WORD =0 LEFT HALF
1301   07441  031717          STA  TEMP1    ADDRESS OF WORD IN TEMP1
1302   07442  121717          LDA  TEMP1,I  GET WORD
1303   07443  074173          SBP  *+3,C    B-REG HAS KEYCODE.BIT 15 IS NOW
1304   07444  050472          AND  MLM1       FLAG FOR WHICH HALF RIGHT-HALF
                                              STORAGE.CLEAR THE LOCATION
1305   07445  067450          JMP  *+3
1306   07446  050345          AND  MASK9    LEFT-HALF STOREAGE.CLEAR THE
                                              LOCATION
1307   07447  074444          SBL  7        LEFT ADJUST
1308   07450  074217          IOR  B        OR INTO WORD
1309   07451  131717          STA  TEMP1.I  STORE WORD IN INPUT BUFFER AREA
1310   07452  067367          JMP  INCP     INCREMENT CP TO NEXT AVAILABLE
                                              LOCATION
1311*
1312   07453  025573  FECH    LDB  CP1      UTILITY ROUTINE USE TO FETCH FROM
1313   07454  063462          JSM  SUB2       INPUT AREA WITH POINTER CP1.
                                              DETERMINE WHICH HALF TO FETCH.
1314   07455  035573          STB  CP1      CP1 HAS BEEN INCREMENT TO NEXT
1315   07456  170402          RET             CHARACTER.(SAME SCHEME OF
                                              ADDRESSING AS ABOVE)
1316   07457  025566  FETCH   LDB  CP       UTILITY ROUTINE USE TO FETCH FROM
1317   07460  063462          JSM  SUB2       INPUT AREA WITH POINTER CP.
1318   87461  067373          JMP  INCP1    ADJUST CP TO POINT TO NEXT
                                              CHARACTER
1319*
1320   07462  074232  SUB2    SBM  FECH1,C  IF BIT15 SET FETCH RIGHT-HALF
1321   07463  074517          LDA  B,I      LEFT-HALF FETCH.GET WORD
1322   07464  070302          SAR  7        RIGHT JUSTIFY
1323   07465  076213          SBP  EXX,S    ADJUST POINTER IN B-REG
1324   07466  074517  FECH1   LDA  B,1      RIGHT-HALF FETCH.GET WORD
1325   07467  050345          AND  MASK9    REMOVE LEFT-HALF
1326   07470  004405          ADB  M1       ADJUST POINTER IN B-REG
1327   07471  170402  EXX     RET
```

```
1329*
1330**********************************
1331*
1332*
1333   07472  021546  DELT   LDA  CMA      UTILITY ROUTINE USED TO DELETE A
1334   07473  031717         STA  TEMP1      PROGRAM LINE POINTED TO BY CMA.
                                             SAVE CURRENT CMA
1335   07474  011542         CPA  END$      ARE WE TRYING TO DELETE BOUNDARY
                                             MARKER
1336   07475  057513         JMP  .AB       YES,DON'T LOOK FOR END OF CURRENT
                                             PROGRAM LINE
1337   07476  070537         LDB  A,I       GET A WORD IN PROGRAM LINE
1338   07477  014405         CPB  M1        IS IT=1? (COULD HAPPEN IN DEFINE-A-KEY)
1339   07500  170402         RET            YES
1340   07501  070070         SIA  *+1       MOVE POINTER
1341   07502  070537         LDB  A,I       GET WORD
1342   07503  075713         SBP  *-2       IF BIT15=1 FOUND BEGINNING OF
                                             NEXT LINE ELSE RECYCLE
1343*
1344   07504  135717  .A4    STB  TEMP1,I   MOVE WORD BACK (ALL OF THE
1345   07505  011542         CPA  END$        PROGRAM IS MOVED BACK) OVER THE
                                             LINE TO BE DELETED. HAS AREG
                                             REACHED END$
                                             YES TRANSFER COMPLETE.
1346   07506  067513         JMP  .A8
1347   07507  045717         ISZ  TEMP1     INCREMENT STOREAGE POINTER
1348   07510  070070         SIA  *+1       INCREMENT FETCH POINTER
1349   07511  070537         LDB  A,I       GET NEXT WORD
1350   07512  067504         JMP  .A4       RECYCLE
1351   07513  021717  .A8    LDA  TEMP1     TEMP1 NOW HAS NEW LOCATION OF
                                             BOUNDARY MARKED
1352   07514  031542         STA  END$      ADJUST END$
1353   07515  066236         JMP  .I8       ADJUST LIM1.NECESSARY IF WORKING
                                             DEFIN.FUNCTION
1354*
1355**********************************
1356*
1357   07516  021705  CKSTP  LDA  .WMOD     UTILITY ROUTINE USE TO CHECK FOR
1358   07517  070273         SAP  *+5,C       STOP ENTRY.THE ROUTINE WILL
                                             ALLOW THE INTERRUPT CLASS OF
                                             KEYCODES
1359   07520  031705         STA  .WMOD
1360   07521  021701         LDA  .WKC      THERE HAS BEEN A NEW KEY ENTRY.
                                             GET KEYCODE
1361   07522  160323         JSM  .SMON,I   ALLOW SYSTEM TO TRY AN EXECUTE IT.
1362   07523  021705         LDA  .WMOD     *MOVE STOP BIT TO SIGN POSITION
1363   07524  070144         SAL  13        *
1364   07525  170402         RET
```

```
1366*
1367*
1368*********************************
1369*
1370*
1371   07526  021705  TRAC   LDA  .WMOD  CHECK FOR TRACE MODE
1372   07527  070211         SLA  INT
1373   07530  063055         JSM  NW1     TRACE MODE
1374   07531  062620         JSM  L5+2    PRINT INPUT
1375   07532  062253         JSM  TSW6
1376   07533  024535  INT    LDB  TOPS    GET ADDRESS OF FIRST WORD IN
                                           COMPILER OUTPUT AREA
1377   07534  035570         STB  ISP     STORE IN POINTER FOR INTERP. WHEN
1378   07535  024414         LDB  AISP      WORKING WITH COMPILER OUTPUT
                                           AREA. PUT ADDRESS OF POINTER B-REG
1379   07536  035605         STB  IMA
1380   07537  065250         JMP  STP
1381*
1382   07540  063135         JSM  DSNO
1383*
1384   07541  021546  UNCP1  LDA  CHA
1385   07542  031663         STA  1663B   SAVE STARTING ADDRESS OF CURRENT
                                           LINE
1386   07543  031662         STA  1662B   SET UP FOR UNCOMPILER
1387   07544  070070         SIA  *+1
1388   07545  070537         LDB  A,I     GET WORD FROM LINE
1389   07546  075653         SBP  *-3
1390   07547  031546         STA  CMA
1391   07550  025663         LDB  1663B   FOUND NEXT LINE; PLACE ADDRESS OF
                                           CURRENT LINE IN B
1392   07551  063716         JSM  RG0     RESET GT0' AND GSB'S ADDRESSES
1393   07552  160311         JSM  UNCPL,I
1394   07553  021573         LDA  CP1
1395   07554  031566         STA  CP
1396   07555  164303         JMP  OUT,I
```

```
1398*
1399*
1400*********************************
1401*
1402   07556  021563          LDA  TSW
1403   07557  025577  EXEQT   LDB  ERST
1404   07560  074452          SBM  EX3+3
1405   07561  070412          SAM  EX3+3   A-REG HAS TSW,IF SIGN BIT SET EOL
1406   07562  063355          JSM  DRECP     ALREADY PRESENT SO RETURN
1407   07563  063457          JSM  FETCH   LOOK AT LAST CHARACTER IN INPUT
                                              STRING
1408   07564  010475          CPA  ENSTM   IS IT ";"?
1409   07565  067601          JMP  EX4       YES,GO RECOMPILE LINE
1410   07566  020437  EX3     LDA  EXEC      NO,LOAD COMPILER EOL CODE
1411   07567  063017          JSM  SKCD    SET UP SKEY, CODE, CN
1412   07570  063100          JSM  PO1     SEND TO COMPILER
1413   07571  074352          SBM  EX1+2   IF -1 COMPILER DIDN'T EXCEPT EOL
1414   07572  025575          LDB  QOTE     CHECK QUOTE FIELD FLAG
1415   07573  074111          SLB  *+2      DO "'S BALANCE?
1416   07574  065317          JMP  1317B   EROOR #1 SYNTAX ERROR.
1417   07575  021563          LDA  TSW
1418   07576  072052  EX1     SAM  *+1,S   SET EOL FLAG IN TSW
1419   07577  031563          STA  TSW      *
1420   07600  170402          RET
1421*
1422   07601  063355  EX4     JSM  DRECP   ; FOUND AT END OF INPUT STRING
1423   07602  063604          JSM  SCAN1   RECOMPILE LINE
1424   07603  067566          JMP  EX3     PLACE EOL IN LINE
```

```
1426*
1427*****************************
1428*
1429  07604  062514  SCAN1  JSM  SCMP    RESET COMPILATION
1430  07605  020354         LDA  UNC1    *SET STORE TO REPLACE MODE
1431  07606  031621         STA  KSTOR   *
1432  07607  021565         LDA  BP      GET BEGINNING POINTER
1433  07610  025566         LDB  CP      GET CURRENT POINTER
1434  07611  031566  SCAN3  STA  CP      MOVE CP BACK TO FIRST CHARACTER
1435  07612  035574         STB  CPE     SAVE VALUE OF CP UPON ENTRY TO
                                           ROUTINE
1436*
1437  07613  063634  .S1     JSM  .S4     CHECK FOR END STRING OR FETCH
                                           KEYCODE
1438  07614  063017         JSM  SKCD
1439  07615  021544         LDA  SKEY
1440  07616  010466         CPA  BGLNE   IS CHARACTER BEGINNING LINE
                                           CHARACTER,BOL
1441  07617  067613         JMP  .S1     YES RECYCLE
1442  07620  063355         JSM  DRECP   RESET POINTER TO INPUT AREA
1443  07621  074742         SBR  16
1444  07622  070250         SZA  .S2     IF CHARACTER ZERO INPUT STRING
                                           COMPLETELY SCANNED
1445  07623  010465         CPA  ENLNE   CHARACTER EOL
1446  07624  067557         JMP  EXEQT     JUMP TO EOL ROUTINE
1447  07625  162202  .S3     JSM  .PRG,I  PROCESS KEYCODE (FORCE KEY TO BE
                                           PROGRAMMABLE)
1448  07626  075253         SBP  .S1     DID SYSTEM EXCEPT IT? IF + YES
1449  07627  170402  .S2     RET            NO
1450*
1451  07630  063457         JSM  FETCH
1452  07631  063376         JSM  .QUOTE+1
1453  07632  010465         CPA  ENLNE
1454  07633  063572         JSM  EX3+4
1455  07634  025566  .S4     LDB  CP
1456  07635  015574         CPB  CPE     IS SCANNING COMPLETE?
1457  07636  067642         JMP  *+4     YES
1458  07637  021705         LDA  .WMOD
1459  07640  071412         SAM  .S4-4   IF NEW KEY RECYCLE UNTILL CP=CPE
1460  07641  067457         JMP  FETCH   NO NEW KEY
1461  07642  055777         DSZ  RSTAK
1462  07643  061253  RET3   JSM  NSTP    USED BY RKBUF AND RPBUF
1463  07644  164303         JMP  OUT,I
1464*
1465  07645  067365         JMP  RET2
1466  07646  160315  RKBUF  JSM  INTRP,I  SETUP RUN TIME STACK
1467  07647  063556         JSM  EXEQT-1   EOL THROUGH SYNTAX
1468  07650  075652         SBM  *-3       IF ERROR CANCELL RETURN
1469  07651  063526         JSM  TRAC      PRINT LINE IF TRACE; SETUP (IMA)→ISP
1470  07652  160524         JSM  XRI,I     CALL INTERPRETER
1471  07653  074752         SBM  RUN3
1472  07654  063523         JSM  CKSTP+5
1473  07655  071312         SAM  RET3      STOP FLAG
1474*
1475  07656  020400  RPBUF  LDA  P4
1476  07657  031563         STA  TSW
1477  07660  063407         JSM  DRCMA+1   MOVE CMA BEGINNING LINE
1478  07661  063516         JSM  CKSTP     ALLOWS INTERRUPT KEYS: SETS UP
                                           .WMOD
1479  07662  071552         SAM  RPBUF-1   IF — STOP FLAG
1480  07663  070704         SAL  2         TRACE BIT IN BIT15
1481  07664  070213         SAP  *+4
```

```
1482  07665  063135          JSM   DSNO      TRACE MODE; GET LINE #
1483  07666  063116          JSM   .STLN
1484  07667  160250          JSM   .PRT1,I
1485  07670  160524  RUN1    JSM   XRI,I     CALL INTERPRETER
1486  07671  067675          JMP   *+4
1487  07672  021563  RUN3    LDA   TSW
1488  07673  000410          ADA   M4
1489  07674  072510          RZA   RUN4
1490  07675  121605          LDA   IMA,I
1491  07676  070073          SAP   *+1,C
1492  07677  031546          STA   CMA
1493  07700  075013          SBP   RPBUF+2   IF + NO ERROR
1494  0770!  063406          JSM   DRCMA     ERROR
1495  07702  160417          JSM   RNTER,I
1496  07703  063045          JSM   NW0
1497  07704  063556          JSM   EXEQT-1
1498  07705  067707          JMP   *+2
1499  07706  062230  RUN4    JSM   CR0       BUILD DISPLAY
1500  07707  061253          JSM   NSTP
1501  07710  066650          JMP   .A7+1
```

```
1503*
1504    07711   000145  RESET   OCT   145
1505*
1506    07712   070742  RGTO    SAR   16
1507    07713   031567          STA   MPH       SET FOR FAST GTO'S
1508    07714   025715          LDB   DFUN
1509    07715   021542          LDA   ENDS
1510    07716   031722  RG0     STA   TEMP2     ENDING   "   "   "
1511*
1512    07717   015722  RG1     CPB   TEMP2     LOOK THROUGH COMPLETE PROGRAM
                                                  FOR GTO'S AND GSB'S
1513    07720   066236          JMP   .I8
1514    07721   074517          LDA   B,I       SET LIM1
1515    07722   050472          AND   MLM1        LEFT HALF
                                                REMOVE BIT 14 AND 15. RIGHT JUSTIFY
                                                  KEYCODE
1516    07723   070302          SAR   7
1517    07724   076052          SBM   *+1,S       UPDATE WORD POINTER
1518    07725   067731          JMP   RG3       JUMP TO CHECK CODE FOR GTO OR GSB
1519*
1520    07726   074517  RG2     LDA   B,I           RIGHT HALF
1521    07727   050345          AND   MASK9     GET RIGHT-HALF CODE
1522    07730   074070          SIB   *+1       UPDATE WORD POINTER
1523*
1524    07731   035723  RG3     STB   TEMP7     SAVE CHARACTER POINTER
1525    07732   063376          JSM   .QOTE+1     CHECK FOR QUOTE FIELD
1526    07733   076410          RZB   *+8       B=1 FOR PRESENTS OF QUOTE FIELD
1527    07734   010350          CPA   GTO       IS CODE GOTO?
1528    07735   067746          JMP   RG4           YES
1529    07736   010526          CPA   GSB       IS CODE GOSUB?
1530    07737   067746          JMP   RG4           YES
1531    07740   025616          LDB   ERASE     CHECK TO SEE OPTION WANTS CONTROL
1532    07741   074110          SZB   *+2       IF ZERO CONTROL STAYS WITH THIS
                                                  ROUTINE
1533    07742   074717          JSM   B,I       ELSE B HAS ADDRESS OF EXPANSION
                                                  ROUTINE
1534    07743   025723          LDB   TEMP7     RETURN CHARACTER POINTER TO
                                                  B-REG.
1535    07744   075132          SBM   RG2,C     PRESENT CODE LEFT-HALF GO FETCH
                                                  RIGHT HALF
1536    07745   067717          JMP   RG1           ELSE GET LEFT HALF
1537*
1538    07746   025723  RG4     LDB   TEMP7     RETURN CHARACTER POINTER TO
                                                  B-REG.
1539    07747   070742          SAR   16
1540    07750   074133          SBP   *+2,C       ADJUST POINTER FOR ZEROING
1541    07751   074070          SIB   *+1           WORD FOLLOWING GTO OR GSB.
1542    07752   074557          STA   B,I       CLEAR WORD
1543    07753   077530  RG6     RIB   RG4-1     NEXT CODE IS IN LEFT-HALF
```

```
1545*
1546*******************************          ORG  3275B      INTERRUPT ROUTINE
1547    03275
1548*
1549*
1550*   IN THE 9820A ONE LEVEL OF INTERRUPT IS
1551*   PROVIDED. FOR THE BASIC SYSTEM THIS LEVEL
1552*   IS OCCUPIED BY THE KEYBOARD. HOWEVER
1553*   BY PROVIDING A R/W SWITCH WORD, THAT
1554*   NORMAL LINKS TO THE KEYBOARD ROUTINE,
1555*   OPTION BLOCKS CAN EXTEND THE INTERRUPT
1556*   HANDLING CAPABILITY OF THE 9820A TO 12 DEVICES
1557*
1558    03275   172700   INTER   SFC  0          IF STOP KEY PROCESS
1559    03276   067302           JMP  SPINT      YES,STOP KEY
1560    03277   031760           STA  SAVEA      SAVE A-REG
1561    03300   020204           LDA  KYBD       ADDRESS OF KEYBOARD(12)
1562    03301   165711           JMP  ILINK,I    NO,ALLOW FOR EXPANSION OF
                                                   INTERRUPT
1563*
1564    03302   031562   SPINT   STA  SAVAS      STOP INTERRUPT;SAVE A-REG
1565    03303   020537           LDA  STOP       LOAD STOP CODE
1566    03304   167305           JMP  *+1,I
1567    03305   007761           DEF  .KBR1      GO SET NEW KEY FLAG
1568    00204           KYBD     EQU  204B
1569*                                        SYSTEM MONITOR ROUTINE (CONT.)
1570    07754                    ORG  RG6+1
1571*
1572    07754   172160   .KBR    OTA  16         ENABLE KEYBOARD TO PLACE CODE ON
                                                   INPUT LINES,RESET
1573    07755   172741           STF  1          LOAD KEYCODE INTO I/O REG
1574    07756   172240           LIA  0          TRANSFER I/O-REG TO A-REG
1575    07757   070342           SAR  8          POSITION DATA TO LSB
1576    07760   050345           AND  MASK9      GET SEVEN BIT CODE
1577    07761   031701   .KBR1   STA  .WKC       SAVE CODE
1578    07762   013711           CPA  RESET      IS ENTRY RESET KEY?
1579    07763   064000           JMP  0B         YES,INITIALIZE MACHINE
1580    07764   070210           SZA  *+4        IF KEYCODE IS NULL IGNORE
1581    07765   021705           LDA  .WMOD
1582    07766   072052           SAM  *+1,S      SET NEW KEY FLAG
1583    07767   031705           STA  .WMOD
1584    07770   021562           LDA  SAVAS      RESTORE A-REG IF STOP INTERRUPT
1585    07771   173760           CLF  16         RESET STOP INTERRUPT
1586    07772   172700           SFC  0          IF STOP DON'T TURN-ON INTERRUPT
1587    07773   170402           RET
1588    07774   021760           LDA  SAVEA      RESTORE A-REG
1589    07775   173741           CLF  1          TURN-ON INTERRUPT
1590    07776   170402           RET
1591*
1592    07777                    ORG  7777B
1593    07777   000000           BSS  1          CHECKSUM WORD
1594*
1595                             END
 * NO ERRORS*
```

```
0001                    ASMB,A,L
0002   00311                ORG  311B
0003   00311  016045        DEF  UNCPL
0004*
0005*
0006   16045                ORG  16045B
0007*
0008*********************************.
0009*
0010*   B-REG HAS THE ADDRESS OF THE LINE          UNCOMPLIER ROUTINE
0011*
0012*********************************
0013*
0014   16045  021545  UNCPL LDA  PSP     ENTRY POINT FOR UNCOMPLIER.B-REG
0015   16046  031573       STA  CP1       POINTER TO LINE TO BE UNCOMPILED.
                                          SET UP POINTER FOR OUTPUT OF
                                          UNCMPLR
0016   16047  072052       SAM  *+1,S     SET BIT15 FOR RIGHT-HALF POSITION
0017   16050  031660       STA  ADD       STORE POINTER,USED TO LOCATE OUT-
                                          PUT CHARACTER.
0018   16051  055663       DSZ  BGSTR     BGSTR USED AS FINISH MARKER FOR
                                          UNCOMPILER
0019*
0020   16052  021625       LDA  1625B     SECURITY CHECK
0021   16053  070610       SZA  LOOP      IF NONZERO SECURE PROGRAM IN
                                          MEMORY
0022   16054  164430       JMP  430B,I    CLEAR USER PROGRAM.
```

```
0024*****************************************
0025*
0026*  RPOL  HAS AN ADDRESS WHICH POINTS TO THE NEXT AVAILABLE WORD
0027*        IN THE REVERSE POLISH STRING.
0028*
0029*  ADD   HAS THE ADDRESS OF WHERE TO STORE THE NEXT SYMBOL OF
0030*        UNCOMPLIED IMFO.
0031*
0032*
0033************************
0034*
0035*     SET UP STACK.
0036*
0037  16055  062200  .I1      JSM  C       CALL ROUTINE TO DUMP OPERATOR
                                            STACK
0038  16056  062305  SETUP   JSM  STNEW   CALL ROUTINE TO STORE CHARACTER IN
0039  16057  020542          LDA  STACK     SKEY IN UNCOMPILER OUTPUT AREA.
0040  16060  031664          STA  POINS   STORE STARTING ADDRESS OF
                                            OPERATOR STACK
0041  16061  020475          LDA  SMCOL
0042  16062  131664          STA  POINS,I INITIALIZE TOP OF STACK WITH ;
0043  16063  070742          SAR  16
0044  16064  031564          STA  IMMFG
0045  16065  031666          STA  POPR    CLEAR FLAG WHICH MARKS THE
                                            OCCURRENCE OF BINARY OP.
0046  16066  031667          STA  OSKP    HOLDS LAST REMOVE OPERATOR FROM
                                            STACK
```

```
0048*
0049*       READY TO START THE LOOP
0050*
0051  16067  062323  LOOP   JSM  NEXTS   ON RETURN A—REG HAS THE SYMBOL
0052  16070  010403         CPA  ONE
0053  16071  062244         JSM  OPTIN
0054  16072  021544         LDA  SKEY
0055  16073  010475  UI     CPA  SMCOL   IS CODE :?
0056  16074  066055         JMP  .I1     YES, DUMP OPERATOR STACK.
                                          INITIALIZE FOR NEW STMT
0057  16075  010437         CPA  EXEC    EOL?
0058  16076  066056         JMP  SETUP   INITIALIZE OPERATOR STACK
0059  16077  010462         CPA  B153    SKIP CODE?
0060  16100  066176         JMP  .SK        YES, SKIP NEXT CODE
0061  16101  010546         CPA  M42     "?
0062  16102  066341         JMP  QUOTE      YES, BUILD QUOTE IN OUTPUT
0063  16103  010507         CPA  B152    "GO TO" OR "GOSUB"
0064  16104  066257         JMP  GOTO       YES, BUILD GTO OR GSB STATEMENT
0065*
0066  16105  014353  S1     CPB  D52     ILLEGAL CODE SKIP IT
0067  16106  066067         JMP  LOOP    GET NEXT CODE
0068  16107  004426         ADB  M13     B—REG HAS CODE CONTROL #
0069  16110  074612         SBM  OPERR   IF CN<13, CODE IS AN OPERATOR
0070  16111  070742         SAR  16      GREATER THEN 13
0071  16112  031666         STA  POPR    ZERO BINARY OPERATOR FLAG
0072  16113  004410         ADB  M4      ADD –4 TO REMAINDER
0073  16114  076310         RZB  *+6
0074*
0075  16115  070742  UOP    SAR  16      CODE IS FOR THE VARIABLES A,B,C,X,Y,Z
0076  16116  031564         STA  IMMFG   CLEAR NUMERIC — IMP * FLAG
0077  16117  062305         JSM  STNEW   PLACE IN ALGEBRAIC STRING
0078  16120  062200         JSM  C       DUMP OPERATOR STACK THRU BINARY
                                          OPERATOR
0079  16121  066067         JMP  LOOP    CONTINUE UNCOMPILING
0080*
0081  16122  074353         SBP  POPER   IF POSITIVE ,REMAINING CODES ARE
                                          FUN OR PROC
0082  16123  066143         JMP  NUMBR   ELSE NUMERIC
0083*
0084  16124  062246  OPERR  JSM  CHECO   CODE IS MOST LIKELY BINARY
                                          OPERATOR. CHECK PRECEDENCE
0085  16125  025666         LDB  POPR    LOAD BINARY FLAG
0086  16126  045666         ISZ  POPR    UPDATE
0087  16127  074150         SZB  *+3     IF NONZERO CHANGE ALGORITHM TO
                                          STACK ( ON EQUALITY
0088  16130  070213         SAP  BB—1
0089  16131  062246  POPER  JSM  CHECO     COMPARE INHAND PRECEDENCE
                                            WITH LAST OPERATOR THEN
0090  16132  070152         SAM  BB      TOP OF STACK. IF INHAND<THEN
                                          STACK, STACK (
0091  16133  070110         SZA  BB        WHICH IS TRUE IF A +
0092*
0093  16134  062235         JSM  INSTK   INSTK PLACES ( IN STACK OUTPUTS )
                                          TO OUTPUT
0094  16135  021561  BB     LDA  CODE
0095  16136  070142         SAR  4
0096  16137  050365         AND  M17     GET INHAND COMPARE PRIORITY
0097  16140  025544         LDB  SKEY    * PLACE OPERATOR (INHAND KEYCODE
                                          IN B—REG)
0098  16141  062276         JSM  STORE   *    IN
0099  16142  066066         JMP  LOOP—1  *    STACK
0100*
0101  16143  025564  NUMBR  LDB  IMMFG
0102  16144  020403         LDA  P1
0103  16145  031564         STA  IMMFG   SET NUMERIC FLAG
```

```
0104   16146  074110          SZB   NUMI     IF ZERØ ,LAST OPERAND ≠ NUMERIC AND
                                                NEXT OPERATOR
0105   16147  062235          JSM   INSTK    ≠ IMP*. IF NONZERO INSERT ( ) AROUND
                                                NUMERIC
0106   16150  062305  NUMI    JSM   STNEW    STORE DIGIT IN OUTPUT
0107   16151  062323          JSM   NEXTS    GET NEXT CODE
0108   16152  014550          CPB   ALPH     END-OF-NUMERIC STRING?
0109   16153  066120          JMP   UOP+3    YES
0110   16154  066150          JMP   NUMI     NO, CONTINUE
0111*
0112*      MOVE THE NEW STRING UP.
0113*
0114   16155  062200  EXIT    JSM   C        EXIT MOVES UNCOMPILER OUTPUT TO
                                                BEGINNING OF INPUT
0115   16156  160547          JSM   FECH,I     STRING AREA. USE CP1 FETCH CODE.
0116   16157  015660          CPB   ADD      B—REG HAS NEXT CODE LOCATION.
                                                ARE WE FINISHED
0117   16160  066166          JMP   *+6        YES, JUMP TO ZERO REMAINDER OF
                                                INPUT AREA.
0118   16161  070137          LDB   A
0119   16162  016437          CPA   EXEC     IS CODE COMPILER EOL?
0120   16163  024465          LDB   ENLNE      YES, CHANGE TO INPUT EOL
0121   16164  161552          JSM   UNCOT,I  SWITCH DIRECTS CODE TO OUTPUT
                                                ROUTINE (MISTOR)
0122   16165  066156          JMP   *–7      RECYCLE
0123   16166  025566          LDB   CP
0124   16167  035573          STB   CP1
0125   16170  015545          CPB   PSP      HAVE WE USED ALL OF THE INPUT AREA
0126   16171  066175          JMP   *+4      YES
0127   16172  074742          SBR   16
0128   16173  161552          JSM   UNCOT,I  ZERO REMAINING AREA IN INPUT
                                                STRING AREA/
0129   16174  066170          JMP   *–4      RECYCLE
0130   16175  170402          RET            RETURN
0131*    SUBROUTINES FOLLOW
0132*
0133   16176  062323  .SK     JSM   NEXTS    SKIP NEXT CODE
0134   16177  066067          JMP   LOOP     RECYCLE
```

(192)

```
0136*
0137*
0138*   THIS SUBROUTINE PULLS OPERATORS OUT OF THE STACK,
0139*     IF THERE IS ANY.
0140*
0141   16200  024542  C      LDB  STACK
0142   16201  074056         CMB
0143   16202  005664         ADB  POINS
0144   16203  075512         SBM  *-6        IF STACK EMPTY RETURN
0145   16204  121664         LDA  POINS,I    LOAD WORD IN TOP OF STACK
0146   16205  055664         DSZ  POINS      MOVE POINTER NEXT LOCATION
0147   16206  070137         LDB  A
0148   16207  074302         SBR  7          RIGHT JUSTIFY STACK PIORITY
0149   16210  035667         STB  OSKP       SAVE IN OLD STACK PIORITY WORD
                                               (OSKP)
0150   16211  050345         AND  RH         CODE IN A--REG
0151   16212  031571         STA  LOPR       SAVE IT
0152   16213  010372         CPA  IMPMT      NO, IF CURRENT OPERATOR IS *(IMP)
                                               DON'T OUTPUT
0153   16214  066231         JMP  C2
0154   16215  062306         JSM  STNEW+1    STORE IN OUTPUT STRING
0155   16216  070742  C1     SAR  16
0156   16217  031564         STA  IMMFG
0157   16220  021571         LDA  LOPR       PUT CODE BACK INTO A--REG
0158   16221  025667         LDB  OSKP       STACK PIORITY BACK IN B--REG
0159   16222  010506         CPA  UNAYM      OPERATORS DUMPED THRU A BINARY
                                               OPERATOR.
0160   16223  066200         JMP  C          CODE WAS UNARY MINUS CONTINUE
                                               UNSTACKING
0161   16224  014370         CPB  M12      IS OPERATOR A FUNCTION
0162   16225  066200         JMP  C          YES CONTINUE DUMPING
0163   16226  014403         CPB  ONE        IS CODE PROCEDURE TYPE. EXAMPLE
                                               PRT
0164   16227  066200         JMP  C          YES CONTINUE DUMPING
0165   16230  164303         JMP  OUT,I
0166*
0167   16231  055564  C2     DSZ  IMMFG
0168   16232  066216         JMP  C1       NUMERIC NOT CAUSING DUMP
0169   16233  020403         LDA  P1
0170   16234  066217         JMP  C1+1      NUMERIC CAUSING DUMP AND IMP*
                                               BEING DUMPED
```

```
0172  16235  020403  INSTK  LDA  ONE      A—REG STACK PIORITY FOR LEFT
                                           PREN.
0173  16236  024503         LDB  LEFT     CODE FOR LEFT PREN.
0174  16237  062276         JSM  STORE    INPUT PRI. WAS LESS THAN STACK
0175  16240  070742         SAR  16
0176  16241  031564         STA  IMMFG
0177  16242  020541         LDA  RIGHT
0178  16243  066306         JMP  STNEW+1  SO PLACE LEFT PREN. IN STACK:
                                           OUTPT RIGHT PREN,
0179*
0180  16244  020405  OPTIN  LDA  M1       CALL OPTION BLOCK SYNTAX
                                           WITH -1 IN A—REG
0181  16245  164551         JMP  OPTN,I
0182*
0183  16246  025667  CHECO  LDB  OSKP     CHECK INHAND COMPARE PRIORITY
                                           WITH LAST STACK PRIORIT
0184  16247  076150         RZB  CHECK+2  IF 0 CHARACTER TO BE COMPARED
                                           IS IN STACK
0185  16250  125664  CHECK  LDB  POINS,I    STACK.
0186  16251  074302         SBR  7         B—REG STACK PIORITY
0187  16252  021561         LDA  CODE
0188  16253  050365  CH     AND  M17       INHAND COMPARE PIORITY IN
0189  16254  070076         TCA              A—REG (STACK -INHAND)
0190  16255  074017         ADA  B
0191  16256  170402         RET
0192*
0193*
0194  16257  062323  GOTO   JSM  NEXTS    GET NEXT CODE
0195  16260  021544         LDA  SKEY
0196  16261  010462         CPA  B153      IS IT SKIP CODE
0197  16262  066274         JMP  GT1         YES, SKIP NEXT CODE
0198  16263  014353         CPB  D52       IGNORE CODE (FOR ZERO
                                             KEYCODE)
0199  16264  066257         JMP  GOTO        YES
0200  16265  014351         CPB  B77       GOTO FLAG FOR OPTION BLOCKS?
0201  16266  066257         JMP  GOTO      YES, DELETE IT
0202  16267  062305         JSM  STNEW       NO, OUTPUT CODE
0203  16270  021544         LDA  SKEY
0204  16271  010475         CPA  SMCOL     WAS IT END-OF-STATEMENT ";"
0205  16272  066057         JMP  SETUP+1     YES, GTO OR GSB UNCOMPILED
0206  16273  066257         JMP  GOTO          NO RECYCLE
0207  16274  062323  GT1    JSM  NEXTS     SKIP NEXT CODE
0208  16275  066257         JMP  GOTO      RECYCLE
0209*
0210*
0211*    SUBROUTINE TO STORE CHAR IN STACK
0212*
0213  16276  014464  STORE  CPB  GOSIN    CHANGE STACK PR. OD GOES INTO
0214  16277  020401         LDA  P3          CHANGE STACK PIORITY TO 3
0215  16300  070444         SAL  7         POSITION PIORITY TO BE INCLUDED
                                             WITH KEYCODE
0216  16301  074217         IOR  B         PLACE PIORITY WITH CODE
0217  16302  045664  .KN    ISZ  POINS     SET NEXT ADDRESS
0218  16303  131664         STA  POINS,I   STORE OPERATOR IN STACK
0219  16304  164303         JMP  OUT,I
```

(194)

```
0221*          SUBROUTINE TO STORE THE NEW STRING
0222*
0223  16305  021544  STNEW  LDA  SKEY    GET CODE TO PLACE IN OUTPUT
0224  16306  010506         CPA  UNAYM   IS IT UNARY MINUS?
0225  16307  020527         LDA  MINUS     YES, CHANGE TO SUBTRACT CODE
0226  16310  025573         LDB  CP1     GET ADDRESS OF NEXT OUTPUT
                                           LOCATION
0227  16311  074333         SBP  *+6,C   WHICH HALF-OF WORD
0228  16312  070444         SAL  7         LEFT-HALF. POSITION CODE
0229  16313  074617         IOR  B,I
0230  16314  074557         STA  B,I     STORE CODE IN OUTPUT
0231  16315  035573         STB  CP1     CP1 HAS BEEN UPDATED SAVE IT
0232  16316  170402         RET
0233  16317  076070         RIB  *+1     RIGHT-HALF
0234  16320  074617         IOR  B,I
0235  16321  074557         STA  B,I     STORE CODE IN OUTPUT
0236  16322  077553         SBP  *-5,S   UPDATE CP1. GO SAVE IT
0237*
0238*
0239*          SUBROUTINE TO GET THE NEXT SYMBOL
0240*
0241  16323  025662  NEXTS  LDB  RPOL    GET ADDRESS OF NEXT CHARACTER
                                           IN POLISH STRING
0242  16324  015663         CPB  BGSTR   HAVE WE TAKEN ALL CODES FROM
                                           LINE?
0243  16325  066337         JMP  EE        YES
0244  16326  074172         SBM  NEX1,C  WHICH HALF OF WORD IS NEXT CODE
0245  16327  074517         LDA  B,I     RIGHT-HALF
0246  16330  076213         SBP  NEX2,S
0247  16331  074517  NEX1   LDA  B,I     LEFT-HALF
0248  16332  070302         SAR  7       RIGHT JUSTIFY CODE
0249  16333  004405         ADB  M1      UPDATE TO NEXT CODE ADDRESS
0250  16334  035662  NEX2   STB  RPOL    SAVE IT
0251  16335  050345         AND  RH      SAVE ONLY SEVEN BIT CODE
0252  16336  164254         JMP  .CS3,I
0253  16337  055777  EE     DSZ  SUB     ALL CODES TAKEN FOR POLISH STRING
0254  16340  066155         JMP  EXIT    ERASE RETURN TO CA ROUTINE.
                                           MOVE UNCOMPILER OUTPUT TO
                                           BEGINNING INPUT
0255*
0256*
0257*          THIS TAKES CARE OF QUOTES
0258*
0259  16341  062305  QUOTE  JSM  STNEW   STORE" IN OUTPUT
0260  16342  062323         JSM  NEXTS     GET NEXT CODE IN QUOTE FIELD
0261  16343  014452         CPB  CNFQ    IS IT " MARKING END OF QUOTE FIELD?
0262  16344  066115         JMP  UOP       YES, TREAT QUOTE FIELD AS AN
                                           OPERAND
0263  16345  066341         JMP  QUOTE   RECYCLE
0264*
0265  00301          WUNSD  EQU  *-UNCPL
```

```
0267*
0268   00254          .CS3     EQU  254B     ADDRESS LINK TO SKCD (SEE CONTROL
                                                SUPERVISOR)
0269   01552          UNCOT    EQU  1552B    R/W LINK WORD. DIRECTS OURPUR OF
                                                UNCOMPILER
0270   01561          CODE     EQU  1561B    16 BIT WORK THAT DIRECTS CODES IN
                                                COMPILING, INTERPRE
0271   01544          SKEY     EQU  1544B    KEYCODE IN PROCESS
0272   01660          ADD      EQU  1660B    END MARKER FOR MOVING
                                                UNCOMPILER OUTPUT
0273   01661          SAVE     EQU  1661B    HOLDS NEXT PROGRAM LINE
0274   01662          RPOL     EQU  1662B    POINTER FOR RETRIEVING CODES FROM
                                                POLISH STRING
0275   01663          BGSTR    EQU  1663B    END MARKER FOR RETREVING CODES
                                                CODES FROM POLISH STRING
0276   01664          POINS    EQU  1664B    OPERATOR STACK POINTER
0277   01665          TEMP2    EQU  1665B    TEMPORARY STOREAGE
0278   01666          POPR     EQU  1666B    FLAG FOR LAST CODE PROCESSED WAS
                                                OPERATOR
0279   01667          OSKP     EQU  1667B    HOLDS STACK PIORITY OF LAST
                                                REMOVED OPERATOR
0280   01571          LOPR     EQU  1571B    SAVES CURRENT OPERATOR BEING
                                                DUMPED
0281   017777         SUB      EQU  1777B    SYSTEM RETURN STACK POINTER
0282   01564          IMMFG    EQU  1564B    NONZERO IF NUMERIC CAUSING DUMP
0283   01565          BP       EQU  1565B    BEGINNING MARKER OF INPUT STRING
                                                AREA
0284   01566          CP       EQU  1566B    NEXT AVAILABLE ADDRESS FOR INPUT
                                                STRING
0285   01545          PSP      EQU  1545B    NEXT AVIALABLE ADDRESS FOR
                                                COMPILER OUTPUT
0286   01573          CP1      EQU  1573B    ADDRESS NEXT CODE IN INPUT STRING
                                                USED WITH FETCH RO
0287   01540          CN       EQU  1540B    CONTROL NUMBER OF CURRENT
                                                KEYCODE
0288   00403          P1       EQU  403B
0289   00372          IMPMT    EQU  372B
0290   00303          OUT      EQU  303B
0291*
0292   00345                   ORG  345B
0293   00345  000177  RH       OCT  177
0294   00351                   ORG  351B
0295   00351  000077  B77      OCT  77
0296   00352  000021  M21      OCT  21
0297   00353  000064  D52      DEC  52
0298   00365                   ORG  365B
0299   00365  000017  M17      OCT  17
0300   00366  000016  M16      OCT  16
0301   00367  000015  M15      OCT  15
0302   00370  000014  M12      OCT  14
0303   00371  000013  M11      OCT  13
0304   00372  000012  P10      OCT  12
0305   00373  000000           BSS  6
0306   00401  000003  P3       OCT  3
0307   00402  000002  TWO      OCT  2
0308   00403  000001  ONE      OCT  1
0309   00404  000000           BSS  1
0310   00405  177777  M1       OCT  177777
0311   00406  177776  M2       DEC  -2
0312   00410                   ORG  410B
0313   00410  177774  M4       DEC  -4
0314   00422                   ORG  422B
0315   00422  177747  MNUM     OCT  177747
0316   00426                   ORG  426B
0317   00426  177763  M13      DEC  -13
```

```
0318   00437                          ORG  427B
0319   00437  000135  EXEC    OCT  135
0320   00452                          ORG  452B
0321   00452  000025  CNFQ    OCT  25
0322   00462                          ORG  462B
0323   00462  000153  B153    OCT  153
0324   00465                          ORG  465B
0325   00465  000141  ENLNE   OCT  141
0326   00466  000140  BGLNE   OCT  140
0327   00475                          ORG  475B
0328   00475  000073  SMCOL   OCT  73
0329   00502                          ORG  502B
0330   00502  177767  M9      DEC  -9
0331   00503  000050  LEFT    OCT  50
0332   00504  000000          BSS  2
0333   00506  000150  UNAYM   OCT  150
0334   00507  000152  B152    OCT  152
0335   00511                          ORG  511B
0336   00511  000154  M154    OCT  154
0337   00527                          ORG  527B
0338   00527  000055  MINUS   OCT  55
0339*       NEW CONSTANTS
0340*
0341   00530                          ORG  530B
0342   00530  00056   ALPH    OCT  56
0343   00532                          ORG  532B
0344   00532  001537  XB      OCT  1537
0345   00535                          ORG  535B
0346   00535  001432  TOPS    OCT  1432
0347   00541                          ORG  541B
0348   00541  000051  RIGHT   OCT  51
0349   00542  001727  STACK   OCT  1727
0350*
0351   00546                          ORG  546B
0352   00546  000042  M42     OCT  42
0353   00464                          ORG  464B
0354   00464  000137  GOSIN   OCT  137
0355   00547                          ORG  547B
0356   00547  000000  FECH    BSS  1
0357*
0358   00551                          ORG  551B
0359   00551  000000  OPTN    BSS  1
0360*
0361*
0362                                  END

*  NO ERRORS*
```

9820 PLOTTER/TYPEWRITTER OPTION BLOCK

```
0001                    ASMB,A,L
0003*
0004   12000                    ORG  12000B
0005*
0006   12000  000001  ID        OCT  1          OPTION BLOCK IDENTIFICATION CODE
0007   12001  000021            DEF  SYNT—ID    RELATIVE ADDRESS OF COMPILE AND
                                                 UNCOMPLIE ROUTINE
0008   12002  000043            DEF  EXEC—ID    RELATIVE ADDRESS OF THE
                                                 EXECUTION ROUTINE
0009   12003  001613            DEF  MNEM—ID    RELATIVE ADDRESS OF THE
                                                 MNEMONIC ROUTINE
0010*
0011*  THE ABOVE IS THE TABLE OF LINKAGE WORDS USED BY THE 9820A
0012*  TO COMMUNICATE WITH THIS OPTION BLOCK.
0013*
0014*  THE RELATIVE KEYCODES ARE AS FOLLOWS:
0015*
0016*    KC=0  TYPE
0017*       1  FORMAT
0018*       2  WRITE
0019*       3  READ
0020*       4  TRANSFER
0021*       5  SCALE
0022*       6  AXES
0023*       7  PEN-UP
0024*       8  LETTER
0025*       9  PLOT
0026*
```

INITIALIZATION ROUTINES.

```
0028*
0029*  GET THE ADDRESS OF RWM WORD POINTING TO THIS BLOCK REM IN
0030*  THE USER AREA. THEN MOVE LIM1 ACCORDINGLY.
0031*
0032*  12004  062110          JSM   L1A     A←ADDRESS OF OPTION BLOCK
                                               DEDICATED R/W WORDS
0033   12005  035716          STB   S       SAVE BASE ADDRESS OF THE OPTION
                                               BLOCK
0034   12006  025543          LDB   LIM1    SAVE PRESENT ADDRESS OF THE FIRST
0035   12007  070577          STB   A,I       WORD OF USER MEMORY IN THE
                                               DEDICATED WORD
0036   12010  004453          ADB   P22     RESERVE 22 WORDS FOR THE I/O BLOCK
0037   12011  035543          STB   LIM1      AND SAVE THE NEW VALUE OR LIM1
0038*
0039*  NEXT SET-UP DEF TABLE FOR THIS BLOCK.
0040*
0041   12012  062070          JSM   .DEF.+1 SET UP LINK WORDS NEEDED FOR THIS
                                               BLOCK
0042*
0043*  DAISY CHAIN THE END LINKAGE.
0044*
0045   12013  021710          LDA   .END2   SAVE PRSENT "END" LINK WORD
0046   12014  131651          STA   DEND,I    IN <DEND>
0047   12015  025716          LDB   S       GET OPTION BLOCK BASE ADDRESS
0048   12016  007700          ADB   INTE    ADD RELATIVE ADDRESS OF OPTION
0049   12017  635710          STB   .END2     BLOCK END ROUTINE AND SAVE IN THE
                                               SYSTEM LINK ADDRESS
0050   12020  17040   L3      RET           RETURN
```

MNEMONIC GENERATION AND SYNTAX CHECK.

```
0052*
0053*  WHEN MNEM IS CALLED, THE A REGISTER CONTAINS THE RELATIVE
0054*  KEYCODE. ON RETURN A WILL BE SET TO N—1 WHERE N IS THE
0055*  NUMBER OF CHARACTERS IN THE MNEMONIC AND THE B REGISTER
0056*  WILL CONTAIN THE CODED MNEMONIC.
0057*
0058*  FIRST GET THE MNEMONIC WORD.
0059*
0060   13613                    ORG  13613B
0061   13613  003617  MNEM      ADA  TAB1     A←RELATIVE KEY CODE+LDB *+1
                                               INSTRUCTION
0062   13614  070016            EXA           EXECUTE THE INSTRUCTION FOUND IN A
0063   13615  020401            LDA  P3       A←3–NUMBER OF CHARACTERS–1
0064   13616  170402            RET           RETURN
0065*
0066   13617  027620  TAB1     LDB  *+1      RELATIVE INSTRUCTION USED TO LOAD
                                               MNEMONIC CODE
0067   13620  051460            DEC  21296    TYP
0068   13621  014664            DEC   6580    FMT
0069   13622  057124            DEC  24148    WRT
0070   13623  044244            DEC  18596    RED
0071   13624  050322            DEC  20690    TFR
0072   13625  046154            DEC  19564    SCL
0073   13626  003405            DEC   1797    AXE
0074   13627  040256            DEC  16558    PEN
0075   13630  031222            DEC  12946    LTR
0076   13631  040624            DEC  16788    PLT

0078*
0079*  FOR BOTH FORWARD AND REVERSE COMPILING, SET CODE="ENT" IF
0080*  CODE=("FMT" OR "RED"). ELSE SET CODE="PRT".
0081*
0082   12021                    ORG  L3+1
0083   12021  021561  SYNT     LDA  CODE     GET SYSTEM CODE WORD
0084   12022  050365            AND  P15      AND SAVE BITS 0—3 TO GET RELATIVE
                                               KEY CODE
0085   12023  024034            LDB  34B      B←SYSTEM TABLE ENTRY FOR PRINT
0086   12024  010401            CPA  P3       A=RELATIVE KEY CODE FOR "RED"?
0087   12025  024031            LDB  31B        YES, B←SYSTEM TABLE ENTRY FOR
                                               ENTER
0088   12026  010403            CPA  P1       A=RELATIVE KEY CODE FOR "FMT"?
0089   12027  024031            LDB  31B        YES, B←SYSTEM TABLE ENTRY FOR
                                               ENTER
0090   12030  035561            STB  CODE     CODE←B–CODE TO BE USED BY THE
                                               SYSTEM
0091   12031  170402            RET           RETURN
```

200

END LINKAGE

```
0093*
0094*  END LINKAGE. DO THE FOLLOWING.
0095*
0096*        SET STANDARD FORMAT.
0097*        DO PEN-UP
0098*
0099   12032  062067  ELINK   JSM   .DEF.   SET UP LINK WORDS NEEDED BY THIS
                                             BLOCK
0100   12033  062110          JSM   L1A     B←BASE ADDRESS OF THE OPTION BLOCK
0101   12034  007701          ADB   SFMT    FORM ABSOLUTE ADDRESS OF THE
                                             STANDARD FORMAT
0102   12035  135650          STB   DFMT,I  <DFMT>←B — ADDRESS OF NEXT
                                             FORMAT TO BE EXECUTED
0103   12036  062717          JSM   L12D    DO A PEN-UP FOR PLOTTER
0104   12037  070742          SAR   16      CLEAR INPUT STATUS BUFFER
0105   12040  131661          STA   DIN,I     AND SAVE IN DIN,I
0106   12041  121651          LDA   DEND,I  GET LINK WORD OF NEXT SEGMENT OF
                                             END ROUTINE
0107   12042  070737          JMP   A,I     JUMP TO THE NEXT SEGMENT OF THE
                                             END ROUTINE
```

EXECUTION BRANCH ROUTINE.

```
0109*
0110*  FIRST SET-UP DEF TABLE FOR OPTION BLOCK.
0111*  EXEC WILL PERFORM A BRANCH TO THE CORRECT EXECUTION ROUTINE
0112*  USING THE RELATIVE KEYCODE AND TAB2.
0113*
0114   12043  062067  EXEC   JSM   .DEF.     SET UP LINK WORDS NEFDED FOR THIS
                                               OPTION BLOCK
0115   12044  021561         LDA   CODE      A←CODE — SYSTEM CODE WORD
0116   12045  050365         AND   P15       SAVE BITS 0—3 — RELATIVE KEY CODE
0117   12046  002053         ADA   TAB2      A←RELATIVE KEY CODE + LDA *+1
                                               INSTRUCTION
0118   12047  070016         EXA             EXECUTE THE INSTRUCTION FOUND IN A
0119   12050  042066         IOR   INST1     A←RELATIVE ADDRESS OF ROUTINE
                                               + JSM ID INSTRUCTION
0120   12051  070016         EXA             EXECUTE INSTRUCTION FOUND IN A
0121   12052  164346         JMP   XJ,I      RETURN TO SYSTEM THROUGH <XJ>
0122*
0123   12053  022054  TAB2   LDA   *+1       INSTRUCTION TO GET RELATIVE
                                               ADDRESS OF ROUTINE
0124   12054  001173         DEF   TY.E—ID   TYPE
0125   12055  000726         DEF   FM.E—ID   FORMAT
0126   12056  001323         DEF   WR.E—ID   WRITE
0127   12057  001352         DEF   RE.E—ID   READ
0128   12060  001564         DEF   CO.E—ID   CONTROL
0129   12061  000136         DEF   SC.E—ID   SCALE
0130   12062  000302         DEF   AX.E—ID   AXES
0131   12063  000717         DEF   PE.E—ID   PEN-UP
0132   12064  000425         DEF   LE.E—ID   LETTER
0133   12065  000205         DEF   PL.E—ID   PLOT
0134   12066  062000  INST1  JSM   ID        INSTRUCTION TO CALL PROPER
                                               SUBROUTINE
```

202

ROUTINE TO SET DEF TABLE

```
0136*
0137*  SET UP DEF'S IN CORE AS FOLLOWS:
0138*
0139*    1650   FORMAT
0140*    1651   END LINK WORD
0141*    1652   X POSITION
0142*    1653   Y POSITION
0143*    1654   LETTER INFORMATION
0144*    1655   BX
0145*    1656   AX
0146*    1657   BY
0147*    1660   AY
0148*    1661   INPUT BUFFER STATUS
0149*
0150  12067  062110  .DEF.  JSM  L1A    A←ADDRESS OF DEDICATED R/W OPTION
                                          BLOCK WORD
0151  12070  070517        LDA  A,I     A←ADDRESS OF R/W MEMORY
                                          RESERVED BY THE OPTION BLOCK
0152  12071  027702        LDB  DLINK   J←ADDRESS OF FIRST WORD WHERE
0153  12072  035663        STB  J         LINK WORDS ARE TO BE STORED
0154  12073  024415        LDB  M10     I←—10
0155  12074  035662        STB  I         NUMBER OF LINK WORDS IN THE TABLE
0156  12075  027651        LDB  K2      B←K2 — CODE WORD FOR 1 OR 4 WORD
                                          STORAGE
0157  12076  066104        JMP  L1E     CONTINUE AT L1E
0158*
0159  12077  045663  L1F   ISZ  J       INCREMENT TABLE ADDRESS
0160  12100  074006        RBR  I       POSITION STORAGE CODE WORD
0161  12101  000403        ADA  P1      INCREMENT R/W MEMORY POINTER
0162  12102  074113        SBP  L1E     LAST PARAMETER NEEDED 4 WORDS?
0163  12103  000401        ADA  P3        YES, ADD 3 MORE TO R/W MEMORY
                                          POINTER
0164  12104  131663  L1E   STA  J,I     SAVE NEW LINK WORD IN THE TABLE
0165  12105  045662        ISZ  I       ALL NINE LINK WORDS FORMED?
0166  12106  066077        JMP  L1F       NO, CONTINUE TO LOOP
0167  12107  170402        RET            YES, RETURN
0168*
0169*  GET ADDRESS OF RWM USER STORAGE BY DOING A SEARCH FOR A
0170*  COMPARISON OF THE ID WORD WITH THOSE FOUND IN THE SYSTEM
0171*  OPTION BLOCK TABLE AREA. WHEN A MATCH IS FOUND, COMPUTE
0172*  THE LINK ADDRESS. SET B WITH ABSOLUTE ADDRESS OF ID.
0173*
0174  12110  024247  L1A   LDB  .TAB7    B←ADDRESS OF TABLE CONTAINING ID
                                          CODES
0175  12111  074517        LDA  B,I     A←OPTION BLOCK ID CODE
0176  12112  050441        AND  K8      SAVE BITS 0—9
0177  12113  012000        CPA  ID      DO THEY MATCH WITH THIS BLOCK?
0178  12114  066116        JMP  *+2       YES, EXIT LOOP
0179  12115  077630        RIB  *-4       NO, INCREMENT ADDRESS AND
                                          CONTINUE TO LOOP
0180*
0181  12116  070076        TCA          A←—A
0182  12117  074417        ADA  B,I     REMOVE ID CODE TO GET BASE
                                          ADDRESS OF THIS BLOCK
0183  12120  070137        LDB  A       B←A
0184  12121  070402        SAR  9       FORM ADDRESS OF DEDICATED R/W
                                          WORDS
0185  12122  000247        ADA  .TAB7     FOR THIS OPTION BLOCK
0186  12123  000400        ADA  P4
0187  12124  170402        RET          RETURN
```

UTILITY ROUTINES

```
0189*
0190*  GET PARAMETER COUNT
0191*
0192   12125  021556  .PAR.   LDA  AP1    A←ADDRESS OF THE ADDRESS OF THE
                                           LAST PARAMETER
                              STA  K      K←A — SAVE POINTER
0193   12126  031664
0194   12127  070076          TCA         A←—A
0195   12130  001557          ADA  AP2    ADD ADDRESS OF THE ADDRESS OF THE
                                           1ST PARAMETER+1
0196   12131  170402          RET         RETURN. A←TOTAL NUMBER OF
                                           PARAMETERS PASSED
0197*
0198*  GET NEXT PARAMETER AND INCREMENT ADDRESS POINTER.
0199*
0200   12132  121664  L2A     LDA  K,I    A←ADDRESS OF THE NEXT PARAMETER
0201   12133  045664          ISZ  K      INCREMENT ADDRESS POINTER
0202   12134  053647          AND  KI     SAVE BITS 0—13
0203   12135  170402          RET         RETURN
0204*
0205*  THE MATH ROUTINES IN THE BASIC MACHINE CAN BE CALLED THROUGH
0206*  THE LINKAGE TABLE FOUND ON THE BASE PAGE AS FOLLOWS:
0207*
0208*      JSM MATH+0,I   <=
0209*                1    =
0210*                2    >
0211*                3    *
0212*                4    /
0213*                5    +
0214*                6    —
0215*
0216*  OP3←(OP1) OPERATOR (OP2)
0217*
```

SCALE EXECUTION FOR 9820A CALCULATOR.

```
0219*
0220*  THE FOLLOWING IS THE SCALE EXECUTION ROUTINE.
0221*
0221*
0222*  GET THE PARAMETER COUNT.
0223*  IF PARAMETER COUNT IS NOT EQUAL TO 4, DO AN ERROR EXIT.
0224*
0225  12136  062125  SC.E   JSM  .PAR.     A←NUMBER OF PARAMETERS PASSED
0226  12137  010400         CPA  P4        IF A=4 THEN
0227  12140  066143         JMP  *+3         SKIP 2 INSTRUCTIONS
0228  12141  024402  L10B   LDB  P2        ELSE SET ERROR 2
0229  12142  065351         JMP  BAD       IMPLIED RETURN
0230*
0231*  SET VALUES OF AY, BY, AX, AND BX FOR USE IN SCALEING
0232*  IN THE PLOT ROUTINE.
0233*
0234  12143  021660         LDA  DAY       I←ADDRESS OF THE Y AXIS
0235  12144  031662         STA  I           SCALE FACTOR
0236  12145  062151         JSM  L10       COMPUTE Y SCALE VALUES
0237  12146  021656         LDA  DAX       I←ADDRESS OF THE X AXIS SCALE
0238  12147  031662         STA  I           FACTOR AND COMPUTE X SCALE
                                             VALUES
0239  12150  066151         JMP  L10       IMPLIED RETURN
0240*
0241*  COMPUTE THE VALUES OF A* AND B* AS FOLLOWS:
0242*
0243*      A*=999/(MAX—MIN)
0244*
0245*      B*=—(A*)*MIN.
0246*
0247  12151  062132  L10    JSM  L2A       A←ADDRESS OF MAX AXIS VALUES
0248  12152  031554         STA  OP1       SAVE ADDRESS IN MATH POINTER OP1
0249  12153  062132         JSM  L2A       A←ADDRESS OF MIN AXIS VALUE
0250  12154  031663         STA  J         SAVE ADDRESS IN J
0251  12155  031553         STA  OP2         AND IN MATH POINTER OP2
0252  12156  021662         LDA  I         A←ADDRESS OF AXIS SCALE FACTOR
0253  12157  031555         STA  OP3         AND SAVE IN MATH POINTER OP3
0254  12160  160273         JSM  MATH+6,I  SUBTRACT MIN VALUE FROM MAX
                                             VALUE
0255*
0256  12161  020477         LDA  .DT1      A←ADDRESS OF TEMPORARY
0257  12162  031554         STA  OP1         REGISTER 1 AND SAVE IN MATH
                                             POINTER OP1
0258  12163  170000         CLR            CLEAR TEMPORARY REGISTER 1
0259  12164  023706         LDA  D9999     STORE 4 BCD 9'S IN THE HIGH ORDER
0260  12165  031741         STA  .T1+1       DIGITS OF TEMPORARY REGISTER 1
0261  12166  020401         LDA  P3        FORM EXPONENT OF+3
0262  12167  070346         RAR  8           AND STORE IN EXPONENT
0263  12170  031740         STA  .T1           PART OF TEMPORARY
                                               REGISTER 1
0264  12171  021662         LDA  I         A←ADDRESS OF AXIS SCALE FACTOR
0265  12172  031553         STA  OP2         AND STORE IN MATH POINTER OP2
0266  12173  160271         JSM  MATH+4,I  DIVIDE 9999 BY (MAX—MIN)
0267*
0268  12174  021663         LDA  J         A←ADDRESS OF MIN VALUE
0269  12175  031554         STA  OP1         AND SAVE IN MATH POINTER OP1
0270  12176  021662         LDA  I         A←I—4
0271  12177  000410         ADA  M4          FORM ADDRESS OF AXIS OFFSET
0272  12200  031662         STA  I             VALUE AND SAVE IN I
0273  12201  031555         STA  OP3             AND MATH POINTER OP3
0274  12202  160270         JSM  MATH+3,I  MULTIPLY MIN VALUE BY SCALE
                                             FACTOR
```

PLOT EXECTION ROUTINE.

```
0278*
0279*  GET THE PARAMETER COUNT AND DO THE FOLLOWING:
0280*
0281*    P=1   GOTO LETTER ROUTINE.
0282*       2  PLOT POINTS X AND Y AFTER SCALING.
0283*    ELSE EXIT WITH ERROR.
0284*
0285   12205  062125  PL.E    JSM  .PAR.    A←NUMBER OF PARAMETERS PASSED
0286   12206  010402          CPA  P2       IF A=2
0287   12207  066213          JMP  L11E       THEN SKIP 2 INSTRUCTIONS
0288   12210  010403          CPA  P1       IF A=1
0289   12211  066445          JMP  LE.EP      THEN GO TO LETTER ROUTINE
0290   12212  066141          JMP  L10B     IMPLIED RETURN
0291*
0292*  SET BOUNDARY INDICATOR AND GET SCALED VALUES OF X AND Y.
0293*
0294   12213  070742  L11E    SAR  16       S16←0
0295   12214  031736          STA  S16        RESET PLOTTER RANGE FLAG
0296   12215  062227          JSM  L11H     SCALE AND CONVERT Y COORDINATE
0297   12216  131653          STA  DYP,I      AND SAVE VALUE IN Y AXIS POSITION
                                               WORD
0298   12217  062233          JSM  L11A     SCALE AND CONVERT X COORDINATE
0299   12220  131652          STA  DXP,I      AND SAVE VALUE IN X AXIS POSITION
                                               WORD
0300*
0301*  IF POINT IS IN RANGE, MOVE PEN TO THE NEW POSITIONS OF X AND Y
0302*  AND PUT PEN DOWN. ELSE LIFT PEN AND THEN MOVE TO THE NEW
0303*  POSITIONS OF X AND Y AND DO NOT PUT THE PEN DOWN.
0304*
0305   12221  021736          LDA  S16      A←S16 — PLOTTER RANGE FLAG
0306   12222  070150          SZA  L11J     IF POINTS ARE IN RANGE (A=0) THEN
0307   1223   062717          JSM  L12D       SKIP 2 INST. ELSE LIFT PEN AND
                                               MOVE TO NEW POSITION
0308   12224  066656          JMP  L12B     IMPLIED RETURN
0309*
0310   12225  062656  L11J    JSM  L12B     MOVE TO NEW POSITION
0311   12226  066713          JMP  L12C     IMPLIED RETURN
0312*
0313*  L11A IS USED TO SCALE THE NEXT PARAMATEK IN THE STACK AND
0314*  MAKE SURE THAT IT IS WITHIN THE RANGE OF 0 TO 9999.
0315*
0316*  S AND S1 ARE USED AS FOLLOWS:
0317*
0318*    S=-2  USE Y SCALE
0319*       -1  USE X SCALE
0320*
0321*   S1=-1      ADD IN OFFSET
0322*      ELSE    DON'T ADD IN OFFSET
0323*
0324*  ON EXIT FROM ROUTINE, S←S-1.
0325*
0326*  FIRST SCALE THE NEXT PARAMATER AS FOLLOWS:
0327*
0328*    (A*)(<AP1>)+(B*)
0329*
0330   12227  020405  L11H    LDA  M1       A←-1
0331   12230  031662          STA  I        I←A — OFFSET COMPUTATION FLAG
0332   12231  020406          LDA  M2       S← -2
0333   12232  031716          STA  S          X/Y COORDINATE FLAG
```

PLOT EXECTION ROUTINE.

```
0334  12233  062132  L11A   JSM   L2A          A←ADDRESS OF DATA WORD TO BE
                                                  CONVERTED
0335  12234  031554         STA   OP1          SAVE ADDRESS IN MATH POINTER OP1
0336  12235  021656         LDA   DAX          A←ADDRESS OF X AXIS SCALE FACTOR
0337  12236  045716         ISZ   S            IF S≠-1
0338  12237  021660         LDA   DAY            THEN A←ADDRESS OF Y AXIS
                                                  SCALE FACTOR
0339  12240  031553         STA   OP2          SAVE ADDRESS OF AXIS SCALE
                                                  FACTOR IN OP2
0340  12241  020477         LDA   .DT1         A←ADDRESS OF TEMPORARY
                                                  REGISTER 1
0341  12242  031555         STA   OP3          SAVE ADDRESS IN MATH POINTER OP3
0342  12243  160270         JSM   MATH+3,I     MULTIPLY COORDINATE BY SCALE
                                                  FACTOR
0343*
0344  12244  045662         ISZ   I            IF I≠-1
0345  12245  066256         JMP   L11G           THEN BY PASS OFFSET COMPUTATION
0346  12246  020477         LDA   .DT1         A←ADDRESS OF TEMPORARY
0347  12247  031554         STA   OP1            REGISTER 1 AND SAVE IN MATH
                                                  POINTER OP1
0348  12250  021655         LDA   DBX          A←ADDRESS OF X AXIS OFFSET
0349  12251  055716         DSZ   S            S←S-1
0350  12252  045716         ISZ   S            IF S≠-1
0351  12253  021657         LDA   DBY            THEN A←ADDRESS OF Y AXIS OFFSET
0352  12254  031553         STA   OP2          SAVE ADDRESS IN MATH POINTER OP2
0353  12255  160272         JSM   MATH+5,I     ADD IN OFFSET TO SCALED VALUE
0354  12256  055662  L11G   DSZ   I            I←I-1
0355*
0356*  CONVERT THE BCD NUMBER TO A BINARY NUMBER. THEN CHECK THE VALUE
0357*  OF THE CONVERTED NUMBER TO MAKE SURE THAT IT IS WITHIN IN THE
0358*  RANGE OF 0 AND 9999. IF THE VALUE IS OUTSIDE OF THE LIMIT,
0359*  SET IT EQUAL TO 0 OR 9999 AS THE CASE MAY BE.
0360*
0361  12257  121555         LDA   OP3,I        GET EXPONENT OF RESULT
0362  12260  070340         AAR   8              AND RIGHT JUSTIFY IT
0363  12261  000410         ADA   M4           IF EXPONENT>3
0364  12262  070613         SAP   L11D           THEN SET LIMIT VALUE AT L11D
0365  12263  021555         LDA   OP3          A←ADDRESS OF SCALED RESULT
0366  12264  061322         JSM   FIXPT        B←BINARY INTEGER VALUE OF
                                                  FLOATING POINT RESULT
0367  12265  074153  L11B   SBP   *+3          IF B<0
0368  12266  045736         ISZ   S16            SET LIMIT FLAG
0369  12267  074742         SBR   16               AND B←0
0370  12270  074117         LDA   B            A←B
0371  12271  007711         ADB   K11          IF B>=10000
0372  12272  074152         SBM   *+3
0373  12273  023707  L11C   LDA   P9999          THEN A←9999
0374  12274  045736         ISZ   S16              AND SET LIMIT FLAG
0375  12275  170402         RET.               RETURN
0376*
0377  12276  070742  L11D   SAR   16           A←0
0378  12277  125555         LDB   OP3,I        B←SIGN OF MANTISSA
0379  12300  075551         SLB   L11C         IF MANTISSA IS PLUS THEN SET HIGH
0380  12301  066274         JMP   L11C+1         LIMIT AT L11C ELSE USE LOW LIMIT
                                                  AND CONTINUE AT L11C+1
```

AXES EXECUTION ROUTINE.

```
0382*
0383*  THE SYNTAX FOR AXES IS AS FOLLOWS:
0384*
0385*     AXES X(ORIGIN),Y(ORIGIN)[,X(TIC),Y(TIC)]
0386*
0387*  TO IMPLIMENT THIS, THE FOLLOWING TABLE HAS BEEN SET UP.
0388*
0389*     S2    CONSTANT AXIS ORIGIN.
0390*     S9    INCREMENT AXIS ORIGIN.
0391*     S10   INCREMENT VALUE.
0392*     S11   ADDRESS LOCATION OF CONSTANT AXIS POSITION.
0393*     S12   ADDRESS LOCATION OF INCREMENT AXIS POSITION.
0394*     S13   HIGH TIC VALUE.
0395*     S14   LOW TIC VALUE.
0396*     S17   OTHER INCREMENT VALUE.
0397*
0398*  SET S10 AND S15 FOR NO INCREMENT.
0399*
0400   12302  023710  AX.E   LDA  K10      S10←S17←10000
0401   12303  031730         STA  S10        SET TIC INCREMENT TO
0402   12304  031737         STA  S17          FULL SCALE VALUE
0403*
0404*  NEXT GET PARAMETER COUNT AND DO THE FOLLOWING.
0405*
0406*     A=2          SKIP TIC MARK SET UP
0407*       4          SET UP TIC MARK VALUES
0408*     ELSE         ERROR MESSAGE.
0409*
0410   12305  062125         JSM  .PAR.    A←NUMBER OF PARAMETERS PASSED
0411   12306  010402         CPA  P2       IF A=2
0412   12307  066324         JMP  L14B       THEN BY PASS TIC MARK SET UP
0413   12310  010400         CPA  P4       IF A=4
0414   12311  066313         JMP  *+2        THEN SET UP TIC MARKS
0415   12312  066141         JMP  L10B     IMPLIED RETURN

0417*
0418*  COMPUTE VALUES FOR TIC MARKS.
0419*
0420   12313  020406         LDA  M2       A←—2 — SET FLAG TO BY PASS OFFSET
                                              COMPUTATION
0421   12314  062230         JSM  L11H+1   A←VALUE OF Y TIC MARK INCREMENT
0422   12315  072110         RZA  *+2      IF A=0
0423   12316  023710         LDA  K10        THEN A←10000
0424   12317  031737         STA  S17      SAVE Y INCREMENT VALUE
0425   12320  062233         JSM  L11A     A←VALUE OF X TIC MARK INCREMENT
0426   12321  072110         RZA  *+2      IF A=0
0427   12322  023710         LDA  K10        THEN A←10000 — FULL SCALE VALUE
0428   12323  031730         STA  S10      SAVE X INCREMENT VALUE
```

AXES EXECUTION ROUTINE.

```
0430*
0431*  COMPUTE VALUES FOR LOCATION OF THE ORIGIN AND IF THEY ARE OUT
0432*  OF RANGE. SET ERROR 21 AND RETURN.
0433*
0434   12324  070742  L14B   SAR  16     S16←0
0435   12325  031736         STA  S16      RESET PLOTTER LIMIT FLAG
0436   12326  062227         JSM  L11H    A←Y VALUE OF COORDINATE ORIGIN
0437   12327  031720         STA  S2      SAVE VALUE IN S2
0438   12330  062233         JSM  L11A    A←X VALUE OF COORDINATE ORIGIN
0439   12331  031727         STA  S9      SAVE VALUE IN S9
0440   12332  021736         LDA  S16     IF S16=0 (COORDINATES ON SCALE)
0441   12333  070150         SZA  *+3       THEN SKIP 2 INSTRUCTIONS
0442   12334  024452         LDB  P21     ELSE SET ERROR 21
0443   12335  065351         JMP  BAD     IMPLIED RETURN
0444*
0445*  SET UP REST OF TABLE AND DRAW X AXIS.
0446*
0447   12336  021653         LDA  DYP     S11←ADDRESS OF Y AXIS PLOTTER
                                           POSITION
0448   12337  031731         STA  S11
0449   12340  021652         LDA  DXP     S12←ADDRESS OF X AXIS PLOTTER
                                           POSITION
0450   12341  031732         STA  S12
0451   12342  062355         JSM  L14A    PLOT X AXIS AND TIC MARKS
0452*
0453*  REARRANGE TABLE AND DRAW Y AXIS AND DO PEN-UP.
0454*
0455   12343  021720         LDA  S2      A←S2 — Y AXIS ORIGIN
0456   12344  025727         LDB  S9      B←S9 — X AXIS ORIGIN
0457   12345  035720         STB  S2      CONSTANT AXIS←B
0458   12346  031727         STA  S9      INCREMENT AXIS←A
0459   12347  021737         LDA  S17     S10←S17
0460   12350  031730         STA  S10       INCREMENT VALUE←X TIC INCREMENT
0461   12351  055731         DSZ  S11     SET X AXIS CONSTANT
0462   12352  045732         ISZ  S12     SET Y AXIS VARIABLE
0463   12353  062355         JSM  L14A    PLOT Y AXIS AND TIC MARKS
0464   12354  066717         JMP  L12D    IMPLIED RETURN

0466
0467*  ROUTINE FOR DRAWNING AXIS.
0468*
0469*  START BY DOING THE FOLLOWING
0470*    1. DO A PEN-UP
0471*    2. MOVE TO 0 OF AXIS TO BE DRAWN.
0472*    3. DO A PEN-DOWN.
0473*
0474   12355  062717  L14A   JSM  L12D    DO A PEN-UP
0475   12356  021720         LDA  S2      <S11>←S2
0476   12357  131731         STA  S11,I     SET CONSTANT AXIS ORIGIN
0477   12360  070742         SAR  16      <S12>←0
0478   12361  131732         STA  S12,I     SET VARIABLE AXIS TO 0 POSITION
0479   12362  062656         JSM  L12A    MOVE PEN TO NEW POSITION
0480   12363  062713         JSM  L12C    PUT THE PEN DOWN
0481*
```

AXES EXECUTION ROUTINE.

```
0482*  SET THE HIGH AND LOW VALUES FOR THE TIC MARKS AND MAKE SURE
0483*  THAT THEY ARE WITHIN BOUNDS.
0484*
0485   12364  025720          LDB  S2      B←S2+47
0486   12365  004356          ADB  P47       HIGH VALUE OF TIC MARK
0487   12366  062265          JSM  L11B    A←VALUE BETWEEN 0<=B<=9999
0488   12367  031733          STA  S13     S13←A — HIGH VALUE OF TIC MARK
0489*
0490   12370  025720          LDB  S2      B←S2—48
0491   12371  007645          ADB  M48       LOW VALUE OF TIC MARK
0492   12372  062265          JSM  L11B    A←VALUE BETWEEN 0<=B<=9999
0493   12373  031734          STA  S14     S14←A — LOW VALUE OF TIC MARK
0494*
0495*  FIND LOCATION OF THE FIRST TIC MARK BY SUBTRACTING THE VALUE
0496*  OF THE INCREMENT FROM THE ORIGIN LOCATION UNTIL THE RESULT
0497*  IS NEGITIVE. THEN ADD THE VALUE OF THE INCREMENT FOR THE
0498*  FIRST TIC MARK.
0499*
0500   12374  021727          LDA  S9      A←S9 — VARIABLE AXIS ORIGIN
0501   12375  025730          LDB  S10     B←S10 — INCREMENT VALUE
0502   12376  074076          TCB          B← −B
0503   12377  074017          ADA  B       WHILE 0<A
0504   12400  071753          SAP  *−1       DO A←A+B
0505*
0506   12401  074076          TCB          B←—B
0507   12402  074017          ADA  B       A←A+B — FIRST POSITIVE TIC MARK
0508*
0509*  MOVE TO COMPUTED LOCATION AND DRAW TIC MARK.
0510*
0511   12403  131732  L14D    STA  S12,I   SAVE VARIABLE AXIS POSITION
0512   12404  062656          JSM  L12B    MOVE PEN TO NEW TIC MARK POSITION
0513   12405  021733          LDA  S13     <S11>←S13
0514   12406  131731          STA  S11,I     CONSTANT AXIS←HIGH VALUE OF TIC
                                              MARK
0515   12407  062656          JSM  L12B    MOVE PEN
0516   12410  021734          LDA  S14     <S11>←S14
0517   12411  131731          STA  S11,I     CONSTANT AXIS←LOW VALUE OF TIC
                                              MARK
0518   12412  062656          JSM  L12B    MOVE PEN
0519   12413  021720          LDA  S2      <S11>←S2
0520   12414  131731          STA  S11,I     CONSTANT AXIS←VARIABLE AXIS
                                              ORIGIN
0521   12415  062656          JSM  L12B    MOVE PEN
0522*
0523*  ADD INCREMENT TO PRESENT POSITION. IF NO OVERFLOW OCCURES,
0524*  MOVE TO NEW COMPUTED POSITION. OTHERWISE, MOVE TO LOCATION
0525*  9999 AND RETURN
0526*
0527   12416  125732          LDB  S12,I   B←(<S12>)+S10
0528   12417  005730          ADB  S10       ADD INCREMENT VALUE TO VARIABLE
                                              AXIS POSITION
0529   12420  062265          JSM  L11B    A←VALUE BETWEEN 0<=B<=9999
0530   12421  075112          SBM  L14D    IF NEW POSITION<10000 THEN
                                              CONTINUE TO LOOP
0531*
0532   12422  023707          LDA  P9999   <S12>←9999
0533   12423  131732          STA  S12,I     MOVE TO END OF AXIS
0534   12424  066656          JMP  L12B    IMPLIED RETURN
```

(210)

LETTER EXECUTION ROUTINES.

```
0536*
0537*  SET LETTER PARAMETERS.
0538*
0539*  GET PARAMETER COUNT. THE SYNTAX FOR LETTER IS AS FOLLOWS:
0540*
0541*      LTR X(CORD),Y(CORD),[,WEIGHT/WIDTH/DIRECTION]
0542*
0543*  THUS IF PARAMETER COUNT IS NOT 2 OR 3, SET ERROR 2 AND RETURN.
0544*
0545  12425  062125  LE.E   JSM  .PAR.  A←NUMBER OF PARAMETERS PASSED
0546  12426  010402         CPA  P2     IF A=2
0547  12427  066437         JMP  L15A     THEN SKIP LETTER FORMAT SET UP
0548  12430  010401         CPA  P3     IF A=3
0549  12431  066433         JMP  *+2    THEN DO LETTER FORMAT SET UP
0550  12432  066141         JMP  L10B   IMPLIED RETURN
0551*
0552*  GET FIRST FOUR DIGITS OF THIRD PARAMETER AND SAVE.
0553*
0554  12433  062132         JSM  L2A    A←ADDRESS OF LETTER FORMAT DATA
                                         WORD
0555  12434  000403         ADA  P1     INCREMENT ADDRESS
0556  12435  070517         LDA  A,I    A←HIGH ORDER DIGITS OF FORMAT
                                         DATA WORD
0557  12436  131654         STA  DLI,I  SAVE LETTER FORMAT
0558*
0559*  COMPUTE THE X AND Y POSITIONS USING THE SCALING ROUTINE AND
0560*  SAVE THE VALUES. DO A PEN-UP AND MOVE TO THE SPECIFIED POSITION.
0561*
0562  12437  062227  L15A    JSM  L11H   A←Y COORDINATE OF NEXT LETTER
0563  12440  131653         STA  DYP,I    AND SAVE POSITION INFORMATION
0564  12441  062233         JSM  L11A   A←X COORDINATE OF NEXT LETTER
0565  12442  131652         STA  DXP,I    AND SAVE POSITION INFORMATION
0566  12443  062717         JSM  L12D   DO PEN UP AND MOVE TO NEW POINT
0567  12444  066656         JMP  L12B   IMPLIED RETURN
```

LETTER EXECUTION ROUTINES.

```
0569*
0570*  EXECUTION OF PLOT "....."
0571*
0572*  SET UP EITHER ALPHA OR NUMERIC LETTER FIELD.
0573*
0574   12445  121664  LE.EP   LDA  K,I      A←ADDRESS OF FIRST PARAMETER
0575   12446  070744         SAL  1         SHIFT A LEFT 1
0576   12447  070272         SAM  ALPHA,C   IF A(15)=1 AND A(15)←0 THEN GO TO ALPHA
0577   12450  070002         SAR  1         SHIFT A RIGHT 1
0578   12451  063270         JSM  L21F      CONVERT NUMERIC PARAMETER AND
                                             SET UP .IBUF
0579   12452  055717         DSZ  S1        DECREMENT CHARACTER COUNT
0580   12453  066456         JMP  ALPHA+2   CONTINUE AT ALPHA+2
0581*
0582   12454  070002  ALPHA  SAR  1         SHIFT A RIGHT 1
0583   12455  062571         JSM  L16E      SET UP PARAMETERS FOR ALPHA STRING
0584*
0585*  SET UP PARAMETERS FOR LETTERING.
0586*
0587   12456  125654         LDB  DLI,I     B←LETTER FORMAT
0588   12457  062564         JSM  L16D      A←HEIGHT INCREMENT VALUE
0589   12460  031730         STA  S10       SAVE VALUE IN S10
0590   12461  062564         JSM  L16D      A←WIDTH INCREMENT VALUE
0591   12462  031731         STA  S11       SAVE VALUE IN S11
0592   12463  074117         LDA  B         FORM DIRECTION INFORMATION
0593   12464  070542         SAR  12          A(0–1)←B(12–13)
0594   12465  050401         AND  P3
0595   12466  070704         SAL  2         FORM 4 BIT SHIFT INFORMATION
0596   12467  000405         ADA  M1          A(0–3)←4*A–1
0597   12470  050365         AND  P15
0598   12471  070544         SAL  5         POSITION SHIFT COUNT
0599   12472  042632         IOR  K4        INCLUDE ROTATE INSTRUCTION
0600   12473  027735         LDB  K9        B←DIRECTION WORD
0601   12474  070016         EXA            EXECUTE INSTRUCTION FOUND IN A
                                             REGISTER
0602   12475  035732         STB  S12       SAVE DIRECTION INFORMATION
0603*
0604*  LIFT PEN
0605*
0606   12476  062717         JSM  L12D      LIFT PEN


0608*
0609*  START NEXT LETTER. SET NEXT LETTER POSITION AND CHECK FOR
0610*  MORE CHARACTERS. IF DONE, EXIT.
0611*
0612   12477  121652  L17A   LDA  DXP,I     S8←CURRENT X POSITION
0613   12500  031726         STA  S8
0614   12501  121653         LDA  DYP,I     S9←CURRENT Y POSITION
0615   12502  031727         STA  S9
0616   12503  045717         ISZ  S1        INCREMENT S1 AND IF S1≠0 (MORE
                                             CHARACTERS LEFT)
0617   12504  066506         JMP  *+2         THEN SKIP 1 INSTRUCTION
0618   12505  170402         RET            ELSE RETURN
0619*
0620*  DO FOLLOWING:
```

212

LETTER EXECUTION ROUTINES.

```
0621*
0622*     1. SET COUNTERS.
0623*     2. GET NEXT CHARACTER.
0624*     3. SET CHARACTER VECTORS.
0625*
0626  12506  020410        LDA  M4     S5←−4
0627  12507  031723        STA  S5       NUMBER OF VECTORS/WORD
0628  12510  070100        AAR  3      S6←−1
0629  12511  031724        STA  S6       SECOND VECTOR WORD FLAG
0630*
0631  12512  062630        JSM  L16G   A←NEXT ASCII CHARACTER
0632*
0633  12513  070706        RAR  15     A←2*A
0634  12514  003671        ADA  K5     ADD (LDB VECTR−64) INSTRUCTION WORD
0635  12515  070016        EXA         EXECUTE INSTRUCTION FOUND IN A
                                         REGISTER
0636  12516  035725        STB  S7     S7←FIRST VECTOR WORD
0637  12517  070070        SIA  *+1    INCREMENT ADDRESS PART OF LOAD
                                         INSTRUCTION
0638  12520  070016        EXA         EXECUTE INSTRUCTION FOUND IN A
                                         REGISTER
0639  12521  035733        STB  S13    S13← SECOND VECTOR WORD
0640*
0641* DRAW NEXT VECTOR.
0642*
0643* SET VALUE OF S3 FOR NEXT HEIGHT SEGMENT. IF S3=−3, CHARACTER
0644* IS FINISHED. EXIT TO L17C. ELSE COMPUTE HEIGHT PART OF NEW
0645* VECTOR.
0646*
0647  12522  062642  L17B  JSM  L16H   A←NEGATIVE INCREMENT COUNT. S3
                                         ALSO SET.
0648  12523  010407        CPA  M3     IF A=−3
0649  12524  066550        JMP  L17C     THEN EXIT LOOP
0650*
0651  12525  021730        LDA  S10    A←HEIGHT INCREMENT VALUE
0652  12526  025732        LDB  S12    B←DIRECTIONAL INFORMATION
0653  12527  074046        RBR  2      POSITION HEIGHT DIRECTION
                                         INFORMATION
0654  12530  062603        JSM  L16F   GENERATE NEXT HEIGHT VECTOR
0655*
0656* COMPUTE WIDTH PART OF NEW VECTOR.
0657*
0658  12531  062642        JSM  L16H   S3←NEGATIVE INCREMENT COUNT
0659  12532  021731        LDA  S11    A←WIDTH INCREMENT VALUE
0660  12533  025732        LDB  S12    B←WIDTH DIRECTION INFORMATION
0661  12534  062603        JSM  L16F   GENERATE NEXT WIDTH VECTOR
0662*
0663* MOVE PEN TO NEW POSITION AND LOWER PEN IF IT IS NOT ALREADY DOWN.
0664*
0665  12535  062656        JSM  L12B   MOVE PEN TO NEW POSITION
0666  12536  062713        JSM  L12C   ELSE PUT PEN DOWN
0667*
0668* CHECK STATUS OF VECTOR WORDS.
0669*
0670  12537  045723        ISZ  S5     INCREMENT S5 AND IF S5≠0 (MORE
                                         VECTORS LEFT IN THE W
0671  12540  066522        JMP  L17B     THEN CONTINUE TO LOOP AT 117B
0672  12541  045724        ISZ  S6     INCREMENTS S6 AND IF S6≠0 (NO MORE
                                         VECTOR WORDS LEFT)
0673  12542  066550        JMP  L17C     THEN EXIT LOOP AND GO TO L17C
0674  12543  020410        LDA  M4     S5← −4
0675  12544  031723        STA  S5       NUMBER OF VECTORS/WORD
0676  12545  021733        LDA  S13    S7←S13
```

LETTER EXECUTION ROUTINES.

```
0677   12546  031725          STA  S7      GET AND SAVE SECOND VECTOR WORD
0678   12547  066522          JMP  L17B    CONTINUE TO LOOP AT L17B
0679*
0680*  CHARACTER FINISHED.
0681*
0682*  LIFT PEN AND COMPUTE NEXT CHARACTER POSITION. DO NEXT CHARACTER.
0683*
0684   12550  062717  L17C    JSM  L12D    LIFT PEN
0685   12551  020407          LDA  M3      S3← −3
0686   12552  031721          STA  S3        INCREMENT COUNT FOR SPACING TO
                                              NEXT CHARACTER
0687   12553  021731          LDA  S11     A←WIDTH INCREMENT VALUE
0688   12554  025732          LDB  S12     B←WIDTH DIRECTION INFORMATION
0689   12555  062603          JSM  L16F    GENERATE WIDTH SPACE VECTOR
0690   12556  070742          SAR  16      S3←0
0691   12557  031721          STA  S3        0 HEIGHT INCREMENT
0692   12560  025732          LDB  S12     B←DIRECTION INFORMATION
0693   12561  074046          RBR  2       POSITION HEIGHT DIRECTION
                                              INFORMATION
0694   12562  062603          JSM  L16F    GENERATE HEIGHT SPACE VECTOR
0695   12563  066477          JMP  L17A    GO TO L17A AND START NEXT
                                              CHARACTER

0697*
0698*  SUBROUTINES USED FOR LETTER.
0699*
0700*  L16D IS USED TO EXTRACT LETTER PARAMETERS FROM THE VECTOR TABLE.
0701*
0702   12564  074546  L16D     RBR  12      A(0—3)←B(12—15)
0703   12565  074117          LDA  B
0704   12566  050365          AND  P15
0705   12567  070544  SAL      5           A←32*A
0706   12570  170402          RET          RETURN
0707*
0708*  L16E SETS UP PARAMETERS FOR ALPHA STRINGS.
0709*  CALLING SEQUENCE:
0710*
0711*      LDA  ADDRESS OF LABEL
0712*      JSM  L16E
0713*      ...  RETURN
0714*
0715*  ON RETURN    S =ADDRESS
0716*               S1=−(COUNT+1)
0717*               S2=52525
0718*
0719*
0720   12571  031716  L16E     STA  S       S←S1←A
0721   12572  031717          STA  S1        SAVE ADDRESS OF ALPHA STRING
0722   12573  020522          LDA  K3      S2←52525B
0723   12574  031720          STA  S2        SET HIGH/LOW POINTER
0724   12575  121717          LDA  S1,I    IF NEXT WORD IN ALPHA STRING<0
0725   12576  070152          SAM  *+3       THEN EXIT LOOP
0726   12577  045717          ISZ  S1      ELSE INCREMENT ADDRESS POINTER
0727   12600  066575          JMP  *−3       AND CONTINUE TO LOOP
0728*
```

214

LETTER EXECUTION ROUTINES.

```
0729  12601  031717          STA  S1     S1←—(NUMBER OF CHARACTERS+1)
0730  12602  170402          RET         RETURN
0731*
0732* L16F IS USED TO COMPUTE THE NEXT VECTOR POSITION.
0733* THE CALLING SEQUENCE IS:
0734*
0735*      LDA  INCREMENT
0736*      LDB  DIRECTION INFORMATION
0737*      JSM  L16F
0738*
0739  12603  074006  L16F   RBR  1      ROTATE B RIGHT 1
0740  12604  074111         SLB  *+2    IF B(0)=1 (NEGATIVE INCREMENT)
0741  12605  070076         TCA           THEN A←—A
0742  12606  031744         STA  AR1    SAVE INCREMENT VALUE
0743  12607  021726         LDA  S8     A←X REFERENCE POSITION
0744  12610  074113         SBP  *+2    IF B(15)=1
0745  12611  021727         LDA  S9       THEN A←Y REFERENCE POSITION
0746  12612  035722         STB  S4     SAVE DIRECTION INFORMATION
0747  12613  055721         DSZ  S3     INCREMENT S3
0748  12614  066616         JMP  *+2    SKIP 1 INSTRUCTION
0749*
0750  12615  001744         ADA  AR1    A←A+AR1 — ADD INCREMENT
0751  12616  045721         ISZ  S3     INCREMENT S3 AND IF S3≠0
0752  12617  066615         JMP  *-2      THEN CONTINUE TO LOOP
0753  12620  070137         LDB  A      B←A
0754  12621  062265         JSM  L11B   A←VALUE BETWEEN 0<=B<=9999
0755  12622  025722         LDB  S4     B←DIRECTION INFORMATION
0756  12623  074153         SBP  *+3    IF B(15)=1
0757  12624  131653         STA  DYP,I    THEN SAVE VALUE IN Y COORDINATE
                                           POSITION
0758  12625  170402         RET         AND RETURN
0759  12626  131652         STA  DXP,I  ELSE SAVE VALUE IN X COORDINATE
                                           POSITION
0760  12627  170402         RET         AND RETURN
0761*
0762* L16G IS USED TO GET THE NEXT CHARACTER FROM THE ALPHA STRING.
0763*
0764  12630  121716  L16G   LDA  S,I    A←WORD FROM ALPHA STRING
0765  12631  025720         LDB  S2     B←HIGH/LOW POINTER
0766  12632  074006  K4     RBR  1      ROTATE POINTER 1 PLACE
0767  12633  035720         STB  S2       AND SAVE POINTER
0768  12634  074153         SBP  *+3    IF B(15)=1 (CHARACTER IN HIGH PART OF
                                           WORD)
0769  12635  070346         RAR  8        THEN ROTATE HALF WAY
0770  12636  066640         JMP  *+2        AND SKIP 1 INSTRUCTION
0771  12637  045716         ISZ  S        ELSE INCREMENT BUFFER ADDRESS
0772  12640  050345         AND  P127   SAVE BITS 0—7
0773  12641  170402         RET         RETURN
0774*
0775* L16H IS USED TO SET-UP THE NEXT DIRECTIONAL VECTOR.
0776*
0777  12642  021725  L16H   LDA  S7     A←VECTOR INFORMATION WORD
0778  12643  070646         RAR  14     ROTATE A 14
0779  12644  031725         STA  S7       AND SAVE VECTOR WORD
0780  12645  050401         AND  P3     SAVE BITS 0—1
0781  12646  070076         ICA         A← −A
0782  12647  031721         STA  S3     S3←—(INCREMENT VALUE −1)
0783  12650  170402         RET         RETURN
```

PLOTTER DRIVERS

```
0785*
0786*  PLOTTER STATUS CHECK
0787*
0788*  REQUEST STATUS AND SET PEN POSITION INFORMATION INTO THE SIGN BIT.
0789*
0790   12651  027677  L12A   LDB  P.SC    B←PLOTTER SELECT CODE
0791   12652  035735         STB  S15     SAVE SELECT CODE IN S15
0792   12653  063555         JSM  L20B+1  GET STATUS
0793   12654  074546         RBR  12      ROTATE A RIGHT 12
0794   12655  170402         RET          RETURN


0796*
0797*  MOVE PLOTTER PEN.
0798*
0799*  SET UP PARAMETERS NEEDED FOR OUTPUT. THEN OUTPUT X AND Y
0800*  POSITIONS TO PLOTTER IN FOUR PASSES. THE BIT FORMAT IS AS
0801*  FOLLOWS:
0802*
0803*    0—3   BCD DECADE 1 OR 100
0804*    4—7   BCD DECADE 10 OR 10000
0805*      8   SYNC/NOT SYNC
0806*      9   BCD/BINARY
0807*     10   UP/DOWN
0808*     11   DATA/PEN
0809*
0810   12656  023677  L12B   LDA  P.SC    S15←PLOTTER SELECT CODE
0811   12657  031735         STA  S15
0812   12660  020410         LDA  M4      I←-4
0813   12661  031662         STA  I         TOTAL NUMBER OF SEGMENTS
0814   12662  020406         LDA  M2      J←-2
0815   12663  031663         STA  J         NUMBER OF SEGMENTS/COORDINATE
0816   12664  121652         LDA  DXP,I   A←X COORDINATE POSITION
0817   12665  072054         SES  *+1,S   E←1
0818   12666  066670         JMP  *+2     SKIP 1 INSTRUCTION
0819*
0820   12667  021664  L13B   LDA  K       A←DATA WORD TO BE OUTPUT
0821   12670  070346         RAR  8       ROTATE WORD HALF WAY
0822   12671  031664         STA  K         AND SAVE DATA WORD
0823   12672  050342         AND  P255    SAVE BITS 0—7
0824   12673  070134         SES  *+2,C   IF E=1 AND E←0
0825   12674  040341         IOR  P256      THEN INCLUDE NON-SYNCH BIT
0826   12675  025662         LDB  I       B(9)←NOT I(0)
0827   12676  074056         CMB            GENERATE BINARY CONTROL BIT
0828   12677  074044         SBL  15          FOR EVERY OTHER OUTPUT WORD
0829   12700  074242         SBR  6
0830   12701  074217         IOR  B       INCLUDE BINARY CONTROL BIT
0831   12702  063535         JSM  L20A    OUTPUT DATA WORD
0832   12703  070075         SEC  *+1,C   E←0
0833*
0834   12704  045662         ISZ  I       INCREMENT I AND IF I≠0 (MORE
                                           SEGMENTS)
0835   12705  066707         JMP  *+2       THEN SKIP 1 INSTRUCTION
                                             AND CONTINUE
0836   12706  170402         RET          ELSE RETURN
```

PLOTTER DRIVERS

```
0837*
0838    12707  045663          ISZ  J        INCREMENT J AND IF J≠0 (NEED 2ND
                                              SEGMENT OF COORDINA
0839    12710  066667          JMP  L13B      THEN LOOP TO OUTPUT NEXT SEGMENT
0840    12711  121653          LDA  DYP.I     A←Y COORDINATE POSITION
0841    12712  066670          JMP  L13B+1    LOOP AND OUTPUT Y SEGMENTS

0843*
0844*   PEN UP/DOWN ROUTINES.
0845*
0846*   FIRST CHECK STATUS AND IF THE PEN IS ALREADY IN THE CORRECT
0847*   POSITION, EXIT THE ROUTINE.
0848*
0849*   PEN DOWN
0850*
0851    12713  062651  L12C    JSM  L12A      A←PLOTTER STATUS
0852    12714  020401          LDA  P3        A←3—PEN DOWN CONTROL BITS
0853    12715  074253          SBP  L13C      IF B(15)=0 (PEN UP) THEN GO TO L13C
0854    12716  170402          RET            ELSE RETURN
0855*
0856*   PEN UP
0857*
0858    12717  062651  L12D    JSM  L12A      A←PLOTTER STATUS
0859    12720  075713          SBP  *-2       IF B(15)=0 (PEN UP) THEN RETURN
0860    12721  020402          LDA  P2        A←2—PEN UP CONTROL BITS
0861    12722  071610  L13C    SZA  L12D-1    IF A=0 (ZERO STATUS) THEN RETURN
0862    12723  070246          RAR  6         POSITION CONTROL BITS
0863    12724  067535          JMP  L20A      OUTPUT PEN COMMAND. IMPLIED
                                              RETURN
0864*
0865    12725  000000          BSS  1         CHECKSUM WORD
```

EQUATES

```
0867*
0868*  EQU'S
0869*
0870   01650        D       EQU  1650R    ADDRESS OF SYSTEM OPTION BLOCK
                                           TABLE
0871   01650        DFMT    EQU  D        LINK WORD TO ADDRESS OF LAST
                                           FORMAT
0872   01651        DEND    EQU  D+1      LINK WORD TO ADDRESS OF NEXT END
                                           ROUTINE SEGMENT
0873   01652        DXP     EQU  D+2      LINK WORD TO X COORDINATE POSITION
0874   01653        DYP     EQU  D+3      LINK WORD TO Y COORDINATE POSITION
0875   01654        DLI     EQU  D+4      LINK WORD TO LETTER FORMAT
                                           INFORMATION
0876   01655        DBX     EQU  D+5      LINK WORD TO X AXIS OFFSET VALUE
0877   01656        DAX     EQU  D+6      LINK WORD TO  X AXIS SCALE FACTOR
0878   01657        DBY     EQU  D+7      LINK WORD TO Y AXIS OFFSET VALUE
0879   01660        DAY     EQU  D+8      LINK WORD TO Y AXIS SCALE FACTOR
0880   01661        DIN     EQU  D+9      LINK WORD TO INPUT STATUS BUFFER
0881   01662        I       EQU  D+10     I
0882   01663        J       EQU  D+11     J
0883   01664        K       EQU  D+12     K
0884   01665        L       EQU  D+13     L
0885*
0886   01716        S       EQU  1716B    S
0887   01717        S1      EQU  S+1      S1
0888   01720        S2      EQU  S+2      S2
0889   01721        S3      EQU  S+3      S3
0890   01722        S4      EQU  S+4      S4
0891   01723        S5      EQU  S+5      S5
0892   01724        S6      EQU  S+6      S6
0893   01725        S7      EQU  S+7      S7
0894   01726        S8      EQU  S+8      S8
0895   01727        S9      EQU  S+9      S9
0896   01730        S10     EQU  S+10     S10
0897   01731        S11     EQU  S+11     S11
0898   01732        S12     EQU  S+12     S12
0899   01733        S13     EQU  S+13     S13
0900   01734        S14     EQU  S+14     S14
0901   01735        S15     EQU  S+15     S15
0902   01736        S16     EQU  S+16     S16
0903   01737        S17     EQU  S+17     S17
0904*
0905   01740        REP.C   EQU  S+18     FORMAT REPEAT COUNT
0906   01741        REP.A   EQU  S+19     FORMAT REPEAT SPECIFICATION
                                           ADDRESS
0907   01742        FW      EQU  S+20     FORMAT FIELD WIDTH
0908   01743        CC      EQU  S+21     OUTPUT CHARACTER COUNT
0909*
0910   01556        AP1     EQU  1556B    ADDRESS POINTER 1
0911   01557        AP2     EQU  1557B    ADDRESS POINTER 2
0912   00346        XJ      EQU  346B     LINK WORD FOR NORMAL RETURN TO
                                           SYSTEM
0913   01554        OP1     EQU  1554B    MATH OPERAND POINTER 1
0914   01553        OP2     EQU  1553B    MATH OPERAND POINTER 2
0915   01555        OP3     EQU  1555B    MATH OPERAND POINTER 3
0916   01543        LIM1    EQU  1543B    POINTER TO FIRST WORD OF USER
                                           MEMORY
0917   01710        .END2   EQU  1710B    LINK WORD USED BY SYSTEM FOR END
                                           ROUTINE
0918   00247        .TAB7   EQU  247B     LINKAGE ATEA FOR OPTION BLOCKS
0919   01740        .T1     EQU  1740B    TEMPORARY REGISTER1
0920   01744        AR1     EQU  1744B    MATH REGISTER 1.
0921   01701        .WKC    EQU  1701B    TEMPORARY STORAGE OF KEYCODE
0922   00265        MATH    EQU  265B     FIRST ENTRY LOCATION OF MATH
                                           ROUTINE TABLE
```

EQUATES

0923	01561	CODE	EQU	1561B	CONTAINS CODE WORD FOR KEYCODE BEING PROCESSED
0924	01351	BAD	EQU	1351B	ROUTINE TO SET ERROR CODE AND ERROR FLAG
0925	00552	SXJ	EQU	552B	ROUTINE TO GET NEXT KEYCODE FORM POLISH STRING
0926	01544	SKEY	EQU	1544B	CONTAINS ACTUAL KEY CODE BEING PROCESSED
0927	00326	RECY	EQU	326B	ROUTINE TO RE-CYCLE PRESENT CODE WORD
0928	01605	IMA	EQU	1605B	LINK WORD TO ADDRESS OF NEXT CODE WORD
0929	00277	.FMT	EQU	277B	LINK WORD TO FORMAT ROUTINE
0930	00554	.IBUF	EQU	554B	ADDRESS OF I/O BUFFER
0931	01703	.WPRT	EQU	1703B	FLOAT N/FIXED N FORMAT WORD
0932	00323	SMON	EQU	323B	LINK WORD TO PASS KEYCODES IN THE ENTER MODE
0933	00312	ENT2	EQU	312B	LINK WORD TO SET UP ENTER ROUTINE
0934	01322	FIXPT	EQU	1322B	ROUTINE TO CONVERT FROM FLOATING TO BINARY INTEGER
0935	12717	PE.E	EQU	L12D	PEN UP EXECUTION ROUTINE
0936	00533	.X3	EQU	533B	DO A RETURN + 1
0937*					
0938*	CONSTANTS				
0939*					
0940	00341	P256	EQU	341B	
0941	00342	P255	EQU	342B	
0942	00345	P127	EQU	345B	
0943	00466	P96	EQU	466B	
0944	00553	P91	EQU	553B	
0945	00505	P90	EQU	505B	
0946	00404	P69	EQU	404B	
0947	00356	P47	EQU	356B	
0948	00530	P46	EQU	530B	
0949	00527	P45	EQU	527B	
0950	00545	P44	EQU	545B	
0951	00544	P43	EQU	544B	
0952	00537	P33	EQU	537B	
0953	00360	P32	EQU	360B	
0954	00453	P22	EQU	453B	
0955	00452	P21	EQU	452B	
0956	00451	P20	EQU	451B	
0957	00352	P17	EQU	352B	
0958	00365	P15	EQU	365B	
0959	00366	P14	EQU	366B	
0960	00367	P13	EQU	367B	
0961	00372	P10	EQU	372B	
0962	00376	P6	EQU	376B	
0963	00400	P4	EQU	400B	
0964	00401	P3	EQU	401B	
0965	00402	P2	EQU	402B	
0966	00403	P1	EQU	403B	
0967	00405	M1	EQU	405B	
0968	00406	M2	EQU	406B	
0969	00407	M3	EQU	407B	
0970	00410	M4	EQU	410B	
0971	00415	M10	EQU	415B	
0972	00420	M16	EQU	420B	
0973	00424	M64	EQU	424B	
0974*					
0975	00522	K3	EQU	522B	
0976	00446	K7	EQU	446B	

```
0977   00441          K8      EQU  441B
0978   00442          K13     EQU  442B

PAGE 0025 #01 EQUATES

0979   00477          .DT1    EQU  477B
```

WRITE EXECUTION ROUTINE

```
1453*
1454*  GET SELECT CODE. IF ERROR OCCURS GO TO THE SELECT CODE ERROR
1455*  ROUTINE. CONTROL WILL GO BACK TO THE SYSTEM.
1456*
1457   13331  045652  L23A    ISZ  DXP     INCREMENT DXP AND IF DXP=0
1458   13332  067334           JMP  *+2       (NO MORE PARAMETERS)
1459   13333  067326           JMP  L23B    THEN GO TO ERROR ROUTINE
1460   13334  063260           JSM  L21E    IF NEXT PARAMETER=ALPHA
1461   13335  067337           JMP  *+2
1462   13336  067326           JMP  L23B      THEN GO TO ERROR ROUTINE
1463*
1464   13337  061322           JSM  FIXPT   B←CONVERTED BINARY INTEGER
                                             DATA WORD
1465   13340  074146  L23C     RBR  4       ROTATE B RIGHT 4
1466   13341  035735           STB  S15     SAVE SELECT CODE
1467   13342  074546           RBR  12      ROTATE B RIGHT 12
1468   13343  004405           ADB  M1      IF SELECT CODE<1
1469   13344  074113           SBP  *+2
1470   13345  067326           JMP  L23B      THEN GO TO ERROR ROUTINE
1471   13346  007655           ADB  M15     IF SELECT CODE>15
1472   13347  074112           SBM  *+2
1473   13350  067326           JMP  L23B      THEN GO TO ERROR ROUTINE
1474   13351  170402           RET          RETURN
```

220

READ EXECUTION ROUTINE

```
1497*
1498*  FIRST GET SELECT CODE AND MAKE SURE THAT Ø<SC<16. ALSO A
1499*  COMMA MUST FOLLOW THE SELECT CODE. IF AN ERROR OCCURS, LOOK
1500*  FOR THE NULL CHARACTER IN THE STRING AND THEN SET THE ERROR
1501*  CODE AND RETURN TO THE SYSTEM.
1502*
1503  13352  070742  RE.E   SAR  16      S1Ø←Ø
1504  13353  031730        STA  S1Ø       INITIALIZE THE NUMBER BUILDER
                                           ROUTINE
1505  13354  160552        JSM  SXJ,I     A←NEXT CHARACTER FROM MEMORY
1506  13355  010352        CPA  P17       IF A=A REFERENCE TO A,B,C,X,Y, OR Z
1507  13356  067501        JMP  L26E        THEN GO TO L26E
1508  13357  063141        JSM  L18C      IF A IS A MUMBER
1509  13360  067354        JMP  *-4           THEN LOOP AND CONTINUE TO
                                             BUILD NUMBER
1510*
1511  13361  010451        CPA  P20       IF A=COMMA
1512  13362  067367        JMP  L26A        THEN SKIP TO L26A
1513*
1514*  ERROR ROUTINE — SEARCH FOR NULL CHARACTER.
1515*
1516  13363  010356  L26Z  CPA  P47       IF A=NULL CHARACTER
1517  13364  067327        JMP  L23B+1      THEN GO TO ERROR ROUTINE
1518  13365  160552        JSM  SXJ,I     ELSE A←NEXT CHARACTER FROM
                                           MEMORY
1519  13366  067363        JMP  L26Z        AND CONTINUE TO LOOP
1520*
1521  13367  025730  L26A  LDB  S1Ø       B←BUILT NUMBER
1522  13370  063340        JSM  L23C      CHECK RANGE OF SELECT CODE
```

PASS DATA FROM INPUT TO OUTPUT DEVICE.

```
1756*
1757*  THE SYNTAX IS AS FOLLOWS:
1758*
1759*   CTR SC1,SC2
1760*
1761*  WHERE SC1=SELECT CODE OF INPUT DEVICE
1762*        SC2=SELECT CODE OF OUTPUT DEVICE
1763*
1764*  GET BOTH SELECT CODES. IF LESS THEN TWO SELECT CODES OR IF
1765*  SELECT CODES OUT OF RANGE, SET ERROR 20 AND RETURN TO SYSTEM.
1766*
1767  13564 063314 CO.E    JSM  L22B   SET UP PARAMETER INFORMATION
1768  13565 063331         JSM  L23A   SET UP SELECT CODE ON INPUT DEVICE
1769  13566 063531         JSM  L24D   SET INPUT STATUS BIT
1770  13567 021735         LDA  S15    S16←S15
1771  13570 031736         STA  S16      TRANSFER SELECT CODE
1772  13571 063331         JSM  L23A   SET UP SELECT CODE OF OUTPUT DEVICE
1773  13572 021735         LDA  S15    S17←S15
1774  13573 031737         STA  S17      TRANSFER SELECT CODE
1775*
1776*  PASS CHARACTER BY CHARACTER FORM THE INPUT DEVICE TO
1777*  THE OUTPUT DEVICE UNTIL AN EOM CODE IS DETECTED
1778*
1779  13574 021736 L27A    LDA  S16    S15←S16
1780  13575 031735         STA  S15      SET INPUT SELECT CODE
1781  13576 063507         JSM  L24B   A←CODE FROM INPUT DEVICE
1782  13577 010401         CPA  P3     IF A=END OF MESSAGE CHARACTER
1783  13600 170402         RET           THEN RETURN WITHOUT PASSING A CR
                                           AND LF
1784  13601 025737         LDB  S17    S15←S17
1785  13602 035735         STB  S15      SET OUTPUT SELECT CODE
1786  13603 063535         JSM  L20A   OUTPUT CODE IN A REGISTER
1787  13604 067574         JMP  L27A   CONTINUE TO LOOP
1788*
1789  13605 000130 P88     DEC  88
```

(222)

VECTER TABLE

```
1791  13612                    ORG 13612B
1792*
1793* LETTER VECTOR TABLE
1794*
1795  13612 140000 VECTR       DEC -16384          BLANK
1796  13613 000000             BSS 15              ENTIRE AREA IS USED BY THE MNEMONIC
                                                   ROUTINE
1797  13632 112034             DEC -27620          (
1798  13633 000136 P94         DEC 94
1799  13634 113034             DEC -27108          )
1800  13635 000044 P36         DEC 36
1801  13636 043226             OCT 43226,16000     ASSIGN
      13637 016000
1802  13640 014524             DEC 6484,27648      +
      13641 066000
1803  13642 002700             DEC 1472            ,
1804  13643 177756 M18         DEC -18
1805  13644 043300             DEC 18112           -
1806  13645 177720 M48         DEC -48
1807  13646 016000             DEC 7168            .
1808  13647 037777 K1          OCT 37777
1809  13650 005300             DEC 2752            /
1810  13651 000740 K2          OCT 740
1811  13652 004242             OCT 4242,5300       0
      13653 005300
1812  13654 014700             DEC 6592            1
1813  13655 177761 M15         DEC -15
1814  13656 105145             OCT 105145,1300     2
      13657 001300
1815  13660 105145             DEC -30107,25100    3
      13661 061014
1816  13662 102152             DEC -31638,11264    4
      13663 026000
1817  13664 124105             OCT 124105,20300    5
      13665 020300
1818  13666 124106             DEC -22458,8268     6
      13667 020114
1819  13670 105034             OCT 105034          7
1820  13671 027512 K5          LDB VECTR-64
1821  13672 044246             DEC 18598,16422     8
      13673 040046
1822  13674 001250             DEC 680,18112       9
      13675 043300
1823  13676 170000 K6          OCT 170000          TYPEWRITER SELECT CODE
1824  13677 160000 P.SC        OCT 160000          PLOTTER SELECT CODE
1825  13700 000032 INTE        DEF ELINK-ID        RELATIVE ADDRESS OF THE OPTION
                                                   BLOCK ROUTINE
1826  13701 001703 SFMT        DEF FMT0-ID         RELATIVE ADDRESS OF THE STANDARD
                                                   FORMAT
1827  13702 001650 DLINK       DEF D               ADDRESS OF OPTION BLOCK R/W AREA
1828  13703 015022 FMT0        OCT 15022,14270,32400  STANDARD FORMAT
      13704 014270
      13705 032400
1829  13706 114631 D9999       DEC -26215
1830  13707 023417 P9999       DEC 9999
1831  13710 023420 K10         DEC 10000
1832  13711 154360 K11         DEC -10000
1833  13712 000000             BSS 2
1834  13714 004242             DEC 2210,25792      A
```

VECTER TABLE

```
      13715  062300
1835  13716  004245
      13717  043040
1836  13720  124501
      13721  026000
1837  13722  004226
      13723  010300
1838  13724  124105
      13725  040054
1839  13726  124105
      13727  040300
1840  13730  124002
      13731  062700
1841  13732  004106
13733 121300
1842  13734  014700
1843  13735  164503  K9
1844  13736  121004
      13737  140000
1845  13740  004112
13741 041300
1846  13742  100054
1847  13743  000000
1848  13744  004132
13745 026000
1849  13746  004052
      13747  140000
1850  13750  004242
      13751  006000
1851  13752  004246
      13753  046000
1852  13754  020212
      13755  022700
1853  13756  004246
      13757  042454
1854  13760  001144
      13761  105300
1855  13762  105221
      13763  140000
1856  13764  100052
13765 140000
1857  13766  100654
1858  13767  006710  ENDE
1859  13770  100122
      13771  126000
1860  13772  005130
      13773  026000
1861  13774  102645
      13775  016000
1862  13776  105002
      13777  140000
1863*
1864
*  NO ERRORS*
```

```
DEC 2213,17952          B
DEC -22207,11264        C
DEC 2198,4288           D
DEC -22459,16428        E
DEC -22459,16576        F
DEC -22526,26048        G
DEC 2118,-23872         H
DEC 6592                I
OCT 164503      PLOTTER LETTER DIRECTION INFORMATION
DEC -24060,-16384       J
DEC 2122,17088          K
DEC -32724              L
BSS 1           CHECKSUM WORD
DEC 2138,11264          M
DEC 2090,-16384         N
DEC 2210,3072           O
DEC 2214,19456          P
DEC 8330,9664           Q
DEC 2214,17708          R
DEC 612,-30016          S
DEC -30063,-16384       T
DEC -32726,-16384       U
DEC -32340              V
DEF 6710B
DEC -32686,-21504       W
DEC 2648,11264          X
DEC -31323,7168         Y
DEC -30206,-16384       Z

END
```

USER-DEFINEABLE FUNCTION BLOCK — — "UDFUN"

```
0001                    ASMB,A,L
0003  14000                    ORG 14000B
0004*
0005* DEFINE-A-KEY OPTION
0006*
0007  14000 000004 ID      DEC 4
0008*
0009  14001 000071         DEF SYNT-ID  BASIC SYSTEM ENTRIES, SYNTAX LINKAGE
0010  14002 001224         DEF EXEC-ID  RUN TIME CODE LINKAGE
0011  14003 000567         DEF MNEM-ID  DISPLAY MNEMONIC
0012*
0013* INITIALIZE
0014*
0015  14004 062765         JSM  FIND    GET ADDRESS OF THIS BLOCK
0016  14005 031720         STA  T2      RETURN WITH ADDRESS IN A-REG.
0017  14006 070402         SAR  9       A = 8,10,12
0018  14007 003724         ADA  NSRW    AND ITS RESERVED R/W
0019*
0020  14010 025543         LDB  LIM1    RESERVE 11 R/W WORDS
0021  14011 070577         STB  A,I      BY MOVING LIM1, A-REG HAS ADDRESS
                                          OF NON-SHARED R/W
0022  14012 004371         ADB  DP11    FOR THIS BLOCK
0023  14013 035543         STB  LIM1    READJUST LIM1
0024*
0025  14014 023120         LDA  DP144   DOUBLE SIZE OF RUN-TIME STACK
0026  14015 031615         STA  BSTK
0027*
0028  14016 062745         JSM  TINK    TRANSFER LINKS. NON-SHARED R/W
                                          TAKEN FROM USER'S R/
0029  14017 021720         LDA  T2
0030  14020 131655         STA  PG,I    SET UP PAGE IDENTIFICATION (ADDRESS
                                          OF OPTION BLOCK)
0031*
0032  14021 003151         ADA  ADND    SET UP ADDRESS OF END LINKAGE ROUTINE
0033  14022 825710         LDB  .END2   GET CURRENT START ADDRESS OF END
                                          LINKAGE
0034  14023 031710         STA  .END2   SAVE NEW ADDRESS, SAVE OLD ADDRESS
                                          FOR DAISY-CHANGIN
0035  14024 135654         STB  EN,I     SAVE OLD ADDRESS FOR DAISY CHAINING
0036*
0037  14025 121655         LDA  PG,I
0038  14026 003427         ADA  ADGT
0039  14027 031623         STA  GOTF    SET UP SPECIAL GTO LINKAGE
0040*
0041  14030 121655         LDA  PG,I
0042  14031 002445         ADA  AZERO   SET UP CONTINUATION OF RGTO IN
0043  14032 031616         STA  ERASE   THE BASIC SYSTEM
0044*
0045  14033 025560         LDB  OSW
0046  14034 121655         LDA  PG,I
0047  14035 002123         ADA  CMPLR   SET UP CONTINUATION OF BASIC SYSTEM
0048  14036 031560         STA  OSW     COMPILER
0049  14037 135656         STB  SOSW,I  SAVE OLD ADDRESS FOR DAISY CHAINING
0050*
0051  14040 121655         LDA  PG,I    CHECK FOR ABSENT BLOCKS
0052  14041 070442         SAR  10
0053  14042 000410         ADA  DM4
0054  14043 131652         STA  NAB,I   NAB = 0,1,2
0055  14044 024431         LDB  OB1     SEARCH THE STARTING ADDRESSES
0056  14045 074517         LDA  B,I
0057  14046 072110         RZA  *+2     IF WORD IS NON-ZERO OPTION BLOCK
                                          IS PRESENT
```

USER-DEFINEABLE FUNCTION BLOCK — — "UDFUN"

```
0058  14047  063050        JSM  ECDS      SET UP EXTRA CODES
0059  14050  024432        LDB  OB2
0060  14051  074517        LDA  B,I
0061  14052  072110        RZA  *+2
0062  14053  063050        JSM  ECDS      SET UP EXTRA CODES
0063  14054  024433        LDB  OB3
0064  14055  074517        LDA  B,I
0065  14056  072310        RZA  *+6
0066  14057  067050        JMP  ECDS      SET UP EXTRA CODES
0067*
0068  14060  121655 ADCD    LDA  PG,I      THIS SUBROUTINE IS USED
0069  14061  070442        SAR  10         TO FETCH A BASE ADDRESS
0070  14062  002065        ADA  *+3       FOR OPTIONAL CODES
0071  14063  070016        EXA
0072  14064  170402        RET
0073  14065  022062        LDA  *-3
0074  14066  000154        OCT  154       BASE ADDRESS OPTION BLOCK #1
0075  14067  000162        OCT  162       BASE ADDRESS OPTION BLOCK #2
0076  14070  000170        OCT  170       BASE ADDRESS OPTION BLOCK #3
```

USER-DEFINEABLE FUNCTION BLOCK —— "UDFUN"

```
0078*  COMPILE
0079*
0080   14071 070113 SYNT   SAP  *+2      ENTRY POINT FOR SYNTAX CHECK
0081   14072 066464        JMP  UNCPL    IF A-REG — REQUST IS FROM UNCOMPILER
0082   14073 023127        LDA  SCODE    IF A+ REQUEST IS FROM SMON
0083   14074 031561 SYNT1  STA  CODE     SET CONTROL NUMBER TO 51
0084   14075 170402        RET
0085*
0086   14076 063071 COPL3  JSM  GETIC    SUBROUTINE USE TO ISOLATE FUNCTION KEYS
0087   14077 031720        STA  T2       SAVE SPECIAL CODE FOR THE KEY
0088   14100 070137        LDB  A
0089   14101 002717        ADA  DM19     IF KEYCODE IS GREATER THAN OR EQUAL
                                           TO 19
0090   14102 071553        SAP  *-5      IT IS A FUNCTION KEY
0091   14103 000372        ADA  DP10
0092   14104 070153        SAP  *+3      KEYCODE IS OPTIONAL KEYS
0093   14105 000377        ADA  DP5
0094   14106 071353        SAP  *-9       KEYCODES ARE FUNCTION KEYS OF THIS BLOCK
0095   14107 164533        JMP  RET1,I   RETURN +1
0096*
0097   14110 121724 CMSET  LDA  EOLFG,I  RESET PORTION OF COMPILER EXTENSION
0098   14111 070310        SZA  CMS1     IF EOLFG IS NON ZERO SPECIAL EOL
                                           REQUIRED
0099   14112 062060        JSM  ADCD
0100   14113 000400        ADA  DP4      SPECIAL EOL IS THE 4TH ADDITIONAL CODE
0101   14114 070137        LDB  A
0102   14115 020537        LDA  COEOL
0103   14116 163742        JSM  CC1,I    SEND SPECIAL EOL TO COMPILER STACK
                                           ROUTINE
0104   14117 070742 CMS1   SAR  16
0105   14120 131724        STA  EOLFG,I  CLEAR SPECIAL EOL FLAG
0106   14121 131651        STA  .G,I     CLEAR SPECIAL DIRECTOR FLAG
0107   14122 066154        JMP  NORET
0108*
0109   14123 000124 CMPLR  DEF  *+1-ID   CONSTANT USED TO SETUP COMPILER
                                           CONTINUATION ROUTIN
0110*
0111   14124 062745 COMPR  JSM  TINK     TRANSFER LINKS TO R/W TEMPORARIES
  12   14125 121655        LDA  PG,I
0113   14126 070442        SAR  10
0114   14127 002711        ADA  TTB0
0115   14130 070016        EXA
0116   14131 131726        STA  IEXC,I   SET-UP CODE USED FOR IMMEDIATE
                                           EXECUTE
0117   14132 025540 CO1    LDB  CN       SET CONTROL NUMBER OF CURRENT CODE
                                           IN B-REG
0118   14133 021601        LDA  STAT
0119   14134 070102        SAR  3
0120   14135 070151        SLA  *+3      BIT 3 OF STAT USED TO ALLOW ";" KEY
0121   14136 014454        CPB  CNFY     BIT 3 SET, IF CURRENT CODE IS ";" SET
                                           ERROR #1
0122   14137 065317        JMP  E1
0123   14140 014355        CPB  CNFR     IS CURRENT CODE CLEAR OR BACKSTACE?
0124   14141 066117        JMP  CMS1     YES, RESET NECESSARY FLAGS
0125   14142 121651        LDA  .G,I     .G IS SPECIAL ROUTING FLAG USED IN
                                           SYNTAX CHECKING
0126   14143 072550        RZA  COPL1    IF NON-ZERO SPECIAL ROUTING OF
                                           CURRENT CODE IS NECES
0127   14144 017131        CPB  CNFO     IS CODE FROM THE OPTION BLOCK?
0128   14145 066161        JMP  COPL2    YES
0129   14146 014457        CPB  CNFE     IS CODE EOL?
0130   14147 066110        JMP  CMSET    YES
0131   14150 017140        CPB  CNFT     IS CODE RETURN?
0132   14151 066426        JMP  ERET     YES, PREFIX WITH SPECIAL RETURN
0133   14152 017141        CPB  CNFD     IS CODE END?
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0134  14153  066434          JMP  EEND     YES, CHECK TO SEE SPECIAL END IS
                                           NECESSARY
0135  14154  121656 NORET    LDA  SOSW,I   RETURN TO OLD COMPILER ENTRY ADDRESS
0136  14155  070737          JMP  A,I
0137*
0138  14156  002542 COPL1    ADA  TTB2     PROCESSING REQUIRES THE USE
0139  14157  070016          EXA           OF .G
0140  14160  070016          EXA
0141*
0142  14161  062076 COPL2    JSM  COPL3    CALL ROUTINE TO ISOLATE KEYCODE TYPE
0143  14162  066341          JMP  EFUN       FUNCTION KEY RETURN
0144  14163  021606          LDA  PS         NON-FUNCTION KEY RETURN, LOAD
                                             PRESENT STATE OF COMP
0145  14164  025720          LDB  T2       RELATIVE KEYCODE
0146  14165  006543          ADB  TTB3     JUMP THROUGH ROUTINE TABLE
0147  14166  074016          EXB
0148  14167  074016          EXB
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0150*
0151  14170 025602 ESCR   LDB  G       "SCR". GET SPECIAL ROUTING WORD BASIC
0152  14171 014376        CPB  DP6     SYSTEM COMPILER.
0153  14172 066421        JMP  GTFUN   IF 6 GTO OR GSB HAS PRECEEDED SCR
0154  14173 163744        JSM  CHKB,I  (BASIC SYSTEM ROUTINE) DOES SCR BEGIN
                                         STATEMENT?
0155  14174 020403        LDA  DP1     YES
0156  14175 131651 ESCR1  STA  .G,I    SET .G TO 1
0157  14176 065304        JMP  CP1     STORE SCR IN COMPILER OUTPUT AREA
0158*
0159  14177 017131 GE1    CPB  CNFO    IS CODE FROM THIS OPTION BLOCK?
0160  14200 066202        JMP  *+2     YES
0161  14201 065317        JMP  E1      ELSE ERROR #1
0162  14202 063071        JSM  GETIC   GET INTERNAL CODE
0163  14203 070150        SZA  *+3     ALLOW IEX
0164  14204 000410        ADA  DM4
0165  14205 071612        SAM  *-4     CODE MUST BE A FUNCTION CODE
0166  14206 020376        LDA  DP6
0167  14207 066175        JMP  ESCR1   SET .G TO 6 AND PLACE CODE IN OUPUT
                                         AREA
0168*
0169  14210 014457 GE6    CPB  CNFE    ENTRY FOR .G=6. IS CODE EOL?
0170  14211 066217        JMP  SNULL   YES
0171  14212 014454        CPB  CNFY    IS CODE EOS?
0172  14213 066217        JMP  SNULL   YES
0173  14214 014451        CPB  CNFC    ALLOW ONLY COMMA
0174  14215 066174        JMP  ESCR1+1 CONTINUE
0175  14216 065317        JMP  E1      SET ERROR #1 FOR ALL OTHER CODES
0176*
0177  14217 020507 SNULL  LDA  NULL
0178  14220 061305        JSM  CP2     STORE SPECIAL CHARACTER FOR
                                         UNCOMPILER
0179  14221 070742 STZG   SAR  16
0180  14222 131651        STA  .G,1    RESET .G
0181  14223 066132        JMP  CO1     CHECK FOR SPECIAL EOL CONDITION
```

```
0183*
0184   14224  163744 EIEX    JSM  CHKB,I  "IEX" DOES IEX START THE STATEMENT
0185   14225  020402         LDA  DP2      YES
0186   14226  066175         JMP  ESCR1    SET .G=2 AND STORE IEX IN OUTPUT
0187*
0188   14227  014457 GE2     CPB  CNFE     ALLOW ONLY EOL OR EOS
0189   14230  066110         JMP  CMSET    TO FOLLOW IEX
0190   14231  014454         CPB  CNFY
0191   14232  066221         JMP  STZG
0192   14233  065317         JMP  E1
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0194*
0195  14234  163744  ECALL.  JSM  CHKB,I  "CALL" DOES CLL BEGIN THE STATEMENT?
0196  14235  020401          LDA  DP3      YES
0197  14236  131651          STA  .G,I     SET .G=3
0198  14237  061304          JSM  CP1      STORE CLL IN OUTPUT
0199  14240  025545          LDB  PSP
0200  14241  074133          SBP  *+2,C
0201  14242  074070          SIB  *+1
0202  14243  070742          SAR  16
0203  14244  074557          STA  B,I      RESERVE NEXT WORD FOR TRANSFER
                                            ADDRESS
0204  14245  074070          SIB  *+1
0205  14246  035545          STB  PSP
0206  14247  065373          JMP  EXIT+9   RETURN TO CONTROL SUPERVISOR
0207*
0208  14250  017131  GE3     CPB  CNFO     .G=3 ENTRY . IS CURRENT CODE FROM THE
                                            OPTION BLOCK
0209  14251  066255          JMP  ALPAR-2  YES
0210  14252  014452          CPB  CNFQ     IS CODE "
0211  14253  066277          JMP  ALQU     YES
0212  14254  065317          JMP  E1       ALL OTHER CODES SYNTAX ERROR #1
0213*
0214  14255  020370          LDA  DP12
0215  14256  061305          JSM  CP2      STORE SPECIAL GTO FLAG IN OUTPUT
0216  14257  063071  ALPAR   JSM  GETIC    GET INTERNAL CODE
0217  14260  000410          ADA  DM4
0218  14261  071552          SAM  *-5      MUST BE FUNCTIO GROUP ELSE ERROR #1
0219  14262  061304  ALP1    JSM  CP1      STORE CURRENT CODE IN OUTPUT
0220  14263  020400          LDA  DP4
0221  14264  031606          STA  PS       SET STATE OF COMPILER TO 4
0222  14265  020366          LDA  DP14
0223  14266  031601          STA  STAT     ALLOW COMMA'S AND QUOTE
0224  14267  070404          SAL  8
0225  14270  141541          IOR  OP,I
0226  14271  131541          STA  OP,I     ALLOW COMMA'S IF PARENTHESIS ARE
                                            USED
0227  14272  145724          ISZ  EOLFG,I  SET SPECIAL EOL FLAG
0228  14273  070742          SAR  16
0229  14274  131651  ALP2    STA  .G,I     RESET .G
0230  14275  020507          LDA  NULL     STORE SPECIAL UNCOMPILER CODE IN
                                            OUTPUT
0231  14276  065305          JMP  CP2      RETURN TO BASIC SYSTEM
0232*
0233  14277  020400  ALQU    LDA  DP4      FIRST CHARACTER WAS A QUOTE
0234  14300  066175          JMP  ESCR1    SET .G=4
0235*
0236  14301  014452  GE4     CPB  CNFQ     END OF LABEL QUOTE?
0237  14302  066262          JMP  ALP1     YES
0328  14303  014454          CPB  CNFY     IF EOS?
0239  14304  065317          JMP  E1       YES, ERROR #1
0240  14305  065304          JMP  CP1      ELSE STORE CURRENT CODE IN OUTPUT
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0242*
0243   14306  061274 ESTRD  JSM  DISIS     "DEF/->F" DISALLOW IMPLIED STORAGE
0244   14307  010401        CPA  DP3       PS=3?
0245   14310  066317        JMP  DEF       YES TREAT KEY AS DEF
0246   14311  073550        RZA  *-5       IF PS IS NOT OPERAND THEN SYNTAX ERROR
0247   14312  020144        LDA  GAZ        TREAT KEY AS ->F
0248   14313  031561        STA  CODE      SET CODE SAME AS ->
0249   14314  070742        SAR  16
0250   14315  031606        STA  PS        SET COMPILER STATE TO THAT OF OPERAND
0251   14316  167741        JMP  CC,I      JUMP TO RASIC SYSTEM STACK ROUTINE
0252*
0253   14317  062060 DEF    JSM  ADCD      "DEF"
0254   14320  000402        ADA  DP2       DEF IS SPECIAL CODE TWO
0255   14321  031544        STA  SKEY
0256   14322  020377        LDA  DP5
0257   14323  066175        JMP  ESCR1     SET .G=5
0258*
0259   14324  017131 GE5    CPB  CNFO      .G=5.  IS CODE FROM THIS OPTION BLOCK?
0260   14325  066327        JMP  *+2       YES, FURTHER PROCESSING NECESSARY
0261   14326  065317        JMP  E1        NO, SYNTAX ERROR #1
0262   14327  063071        JSM  GETIC
0263   14330  000410        ADA  DM4       ALLOW ONLY FUNCTION KEY
0264   14331  071652        SAM  *-3       IF NOT FUNCTION KEY, SYNTAX ERROR
0265   14332  061304        JSM  CP1       STORE KEY IN OUTPUT
0266   14333  020402        LDA  DP2       SET .G=2
0267   14334  066274        JMP  ALP2      STORE NULL CHARACTER FOR UNCOMPILER
0268*
0269   14335  020077 EPAR   LDA  REG       "P"
0270   14336  031561        STA  CODE      SET P TO BE CHECKED BY BASIC SYSTEM
0271   14337  163760        JSM  SKCD1,I   SAME AS THE R KEY.
0272   14340  164314        JMP  BCPLR,I   RETURN TO BASIC COMPILER
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0274*
0275  14341 021602 EFUN   LDA G        "F*,G*,H*". GET SPECIAL ROUTING FLAG
0276  14342 010376        CPA DP6      OF BASIC SYSTEM. IS FUNCTION KEY
                                         FOLLOWING GTO OR G
0277  14343 066421        JMP GTFUN    YES
0278  14344 021720        LDA T2       GET INTERNAL CODE
0279  14345 062622        JSM SFFN     LOOK TO SEE IF FUNCTION DEFINED
0280  14346 074652        SBM EF1      IF B- FUNCTION NOT DEFINED
0281  14347 004406        ADB DM2      FUNCTION DEFINED. B-REG HAS ADDRESS
                                         OF START
0282  14350 074517        LDA B,I      LOOK AT ID WORD
0283  14351 070113        SAP *+2
0284  14352 066374        JMP FNCLL    BIT 15 SET THEREFORE PROGRAM IS A
                                         FUNCTION
0285*
0286  14353 004402        ADB DP2      PROGRAM IS A SUBROUTINE. CALLED
                                         WITHOUT GSB, GTO OR
0287  14354 121653        LDA IXFLG,I  GET IMMEDIATE EXECUTE FLAG
0288  14355 070110        SZA *+2      IS IT SET?
0289  14356 067454        JMP CIEX     YES, EXECUTE PROGRAM
0290  14357 163747        JSM GAS2,I   BASIC ROUTINE TO FETCH CODES
                                         (B-REG ADDRESS)
0291  14360 050345        AND RH
0292  14361 111726        CPA IEXC,I   IS FIRST CODE IMMEDIATE EXECUTE?
0293  14362 067454        JMP CIEX     YES, THEN EXECUTE PROGRAM
0294*
0295  14363 021606 EF1    LDA PS
0296  14364 163744        JSM CHKB,I   IS FUNCTION FIRST KEY OF STATEMENT
0297  14365 021602        LDA G        YES
0298  14366 010374        CPA DP8      IF G=8 ONLY EOL OR EOS IS CORRECT
0299  14367 065317        JMP E1       TRUE
0300  14370 023730        LDA GSBK     PLACE GSB BEFORE FUNCTION KEY
0301  14371 061305        JSM CP2      STORE GSB IN OUTPUT
0302  14372 062240        JSM ECALL+4  SET A BLANK WORD FOR THE FAST-ACCESS
0303  14373 066421        JMP GTFUN
0304*
0305  14374 020141 FNCLL  LDA SQRT     FUNCTION KEY IS TO BE TREATED AS A
                                         FUNCTION
0306  14375 031561        STA CODE     SET CODE WITH PRIORITIES AS SQUARE-
                                         ROOT
0307  14376 070142        SAR 4
0308  14377 070204        SAL 12
0309  14400 031720        STA T2       SET STACK PRIORITY IN PROPER
                                         POSITION FOR SPECIAL CO
0310  14401 163760        JSM SKCD1,I  BASIC SYSTEM ROUTINE TO SET-UP CN,
                                         AND SKEY
0311  14402 024407        LDB DM3
0312  14403 035721        STB T3       COUNTER FOR LOOP BELOW
0313  14404 160314        JSM BCPLR,I  SEND FUNCTION KEY TO BASIC COMPILER
0314  14405 021720        LDA T2       SET-UP BLANK WORD FOR THE FAST-
                                         ACCESS
0315  14406 163743        JSM STSK,I   STACK OPERATOR WITH CODE=0
0316  14407 045721        ISZ T3       THREE CODES STACKED?
0317  14410 066405        JMP *-3      NO
0318  14411 062060        JSM ADCD     YES
0319  14412 000377        ADA DP5      SPECIAL CODE FOR FUNCTION TERMINATION;
                                         "FUNC"
0320  14413 040446        IOR ALCOM
0321  14414 041720        IOR T2       ALL COMMAS WITHIN THE FIRST LEVEL OF
                                         PARENTHESIS
0322  14415 163743        JSM STSK,I   STACK SPECIAL CODE
0323  14416 062060        JSM ADCD
0324  14417 000376        ADA DP6      SPECIAL CHARACTER TO BEGIN FUNCTION
                                         STRING: "TAU"
```

```
0325  14420  065305          JMP  CP2      STORE IT IN OUTPUT AREA
0326*
0327  14421  020374  GTFUN   LDA  DP8      SET BASIC SYSTEM SPECIAL ROUTING
                                             FLAG
0328  14422  031602          STA  G        TO 8.
0329  14423  020370          LDA  DP12     SPECIAL GOTO FLAG FOR BASIC SYSTEM
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0330  14424  061305          JSM  CP2      STORE IT IN OUTPUT
0331  14425  065304          JMP  CP1      STORE FUNCTION KEY IN THE OUTPUT
0332*
0333  14426  022551  ERET    LDA  033      "RET"
0334  14427  160254          JSM  SKCD,I   SET UP CN
0335  14430  062060          JSM  ADCD
0336  14431  000403          ADA  DP1      SPECIAL RETURN IS THE FIRST
0337  14432  031544          STA  SKEY     ADDITION CODE FOR THE OPTION
                                             BLOCK
0338  14433  066154          JMP  NORET    ALLOW BASIC SYSTEM TO CHECK
                                             SYNTAX
0339*
0340  14434  021546  EEND    LDA  CMA      "END" GET PROGRAM STORAGE
                                             POINTER
0341  14435  070076          TCA
0342  14436  001543          ADA  LIM1     ADD IN ADDRESS OF BEGINNING OF
                                             MAINLINE
0343  14437  070113          SAP  *+2      IF RESULT IS + IN PROTECTED
                                             SUBPROGRAM AREA
0344  14440  066154          JMP  NORET    IF — END IN MAINLINE: NORMAL
                                             RETURN
0345  14441  062060          JSM  ADCD
0346  14442  000401          ADA  DP3      SPECIAL END IS THIRD CODE OF
                                             ADDITIONAL
0347  14443  061305          JSM  CP2      CODES FOR OPTION BLOCK
0348  14444  066154          JMP  NORET
```

(284)

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0350*
0351   14445  000446  AZERO  DEF  *+1-ID     THIS ROUTINE IS AN EXTENSION OF
                                                THE RESET
0352   14446  160254         JSM  SKCD,I     OF FAST-ACCESS WORDS OF THE BASIC
                                                MACHINE:
0353   14447  010403         CPA  DP1        RGTO.
0354   14450  066452         JMP  *+2         THIS LOOKS FOR THE WORDS
                                                FOLLOWING THE
0355   14451  066460         JMP  AZ1        FUNCTION CODES AND CLL.
0356   14452  062745         JSM  TINK
0357   14453  063071         JSM  GETIC
0358   14454  010403         CPA  DP1
0359   14455  066462         JMP  AZ2
0360   14456  010367         CPA  DP13
0361   14457  066462         JMP  AZ2
0362   14460  021544  AZ1    LDA  SKEY
0363   14461  170402         RET
0364*
0365   14462  055777  AZ2    DSZ  RSTAK      NOT OPTION BLOCK CODE
0366   14463  167766         JMP  RG4,I      SPECIAL RETURN TO BASIC SYSTEM
                                                ROUTINE
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0368*
0369   14464  062745  UNCPL  JSM  TINK     THIS ROUTINE IS FOR UNCOMPILING
                                            STATEMENT
0370   14465  063071         JSM  GETIC    COMPOSED OF KEYCODES FROM
                                            THIS OPTION BLOCK
                              CPA  DP2      CODE -> F?
0371   14466  010402         JMP  UGAZ     YES
0372   14467  066520         CPA  DP3      CODE P?
0373   14470  010401         JMP  UPAR     YES
0374   14471  066514         CPA  DP9      CODE SPECIAL RETURN?
0375   14472  010373         JMP  PROC     YES
0376   14473  066505         CPA  DP10     CODE DEF?
0377   14474  010372         JMP  PROC     YES
0378   14475  066505         CPA  DP11     CODE SPECIAL END?
0379   14476  010371         JMP  USEOL    YES
0380   14477  066516         CPA  DP12     CODE SPECIAL EOL?
0381   14500  010370         JMP  USEOL    YES
0382   14501  066516         CPA  DP14     CODE FOR TAU?
0383   14502  010366         JMP  USEOL    YES
0384   14503  066516         RZA  UFUN     ELSE NON-ZERO CODES F*, G*, OR H*
0395   14504  072150         LDA  CLOD     CODE IEX
0386   14505  020012  PROC   JMP  SYNT1    SET CODE = LOD
0387   14506  066074
0388*
0389   14507  163774  UFUN   JSM  NEXTS,I  UNCOMPILE FUNCTION CALL
0390   14510  021544         LDA  SKEY
0391   14511  071710         SZA  *-2      LOOK FOR F*, G*, OR H*
0392   14512  021717         LDA  T1       FOUND IT
0393   14513  031544         STA  SKEY
0394   14514  020141  UPAR   LDA  SORT     SET CODE FOR FUNCTION (SQUARE
                                            ROOT)
0395   14515  066074         JMP  SYNT1
0396*
0397   14516  020005  USEOL  LDA  IGN      SET CODE FOR IGNORE
0398   14517  066074         JMP  SYNT1
0399*
0400   14520  055777  UGAZ   DSZ  RSTAK    -> UNCOMPILING,ERASE RETURN TO
                                            OPTION
0401   14521  055777         DSZ  RSTAK    BLOCK SYNTAX CALLING ROUTINE;
                                            RETURN TO UNCOMPILER
0402   14522  020144         LDA  GAZ      THE SINGLE KEYCODE IS TREATED
                                            AS TWO
                              STA  CODE     CODES FOR UNCOMPILING, F->
0403   14523  031561         JSM  CHECO,I  CHECK FOR INSERTING OF
0404   14524  163773                       PARENTHESIS
0405   14525  025666         LDB  POPR     IF LAST OPERATOR WAS BINARY
0406   14526  045666         ISZ  POPR     CHECK FOR EQUALITY OF PRIORITIES
0407   14527  074150         SZB  *+3
0408   14530  070213         SAP  BB-1     (WORD NOT NECESSARY)
0409   14531  163773         JSM  CHECO,I  IF IN HAND < THEN STACK
0410   14532  070152         SAM  BB
0411   14533  070110         SZA  BB
0412*
0413   14534  163772         JSM  INSTK,I  INSERT PARENTHESIS
0414   14535  020401  BB     LDA  DP3
0415   14536  025544         LDB  SKEY     STORE -> IN OPERATOR STACK
0416   14537  163771         JSM  STORE,I  DUMP STACK FOR F (EASIEST WHY TO
0417   14540  163770         JSM  C,I      KEEP UNCOMPILER)
                                            CORRECT): RETURN TO BASIC
0418   14541  167767         JMP  LOOP,I   UNCOMPILER
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0420*
0421*****************************************
0422*
0423*  COMPILER TABLE
0424*
0425    14542  022560  TTB2    LDA  TTB4-1
0425*
0427    14543  026545  TTB3    LDB  *+2
0428    14544  066170          JMP  ESCR     -1
0429    14545  066224          JMP  EIEX     0
0430    14546  066234          JMP  ECALL    1
0431    14547  066306          JMP  ESTRD    2
0432    14550  066335          JMP  EPAR     3
0433    14551  000033  O33     OCT  33
0434    14552  016000  PGM     OCT  1600
0435    14553  000406  ADM2    DEF  406B
0436    14554  001625  ECRW    DEF  1625B
0437    14555  001626  OBRW    DEF  1626B
0438    14556  066426          JMP  ERET     9
0439    14557  066306          JMP  ESTRD    10
0440    14560  066434          JMP  EEND     11
0441*
0442    14561  066177  TTB4    JMP  GE1
0443    14562  066227          JMP  GE2
0444    14563  066250          JMP  DE3
0445    14564  066301          JMP  GE4
0446    14565  066324          JMP  GE5
0447    14566  066210          JMP  GE6
```

USER-DEFINEABLE FUNCTION BLOCK —"UDFUN"

```
0449*
0450*  SUPPLY MNEMONIC INFO
0451*
0452   14567  062745  MNEM   JSM  TINK    ROUTINE USE TO RETURN MNEMONIC
0453   14570  063071         JSM  GETIC    TO BASIC SYSTEM, GET INTERNAL
                                            CODE
0454   14571  010401         CPA  DP3      IS CODE P?
0455   14572  066662         JMP  MNE1     YES
0456*
0457   14573  062622         JSM  SFFN     IF FUNCTION KEY FIND ROUTINE
0458   14574  074212         SBM  *+4      IF B- ROUTINE NOT DEFINED
0459*
0460   14575  163747         JSM  GAS2,I   ROUTINE DEFINED LOOK TO SEE IF
0461   14576  010517         CPA  BQT      FIRST STATEMENT IS LABEL?
0462   14577  066601         JMP  *+2      YES, MNEMONIC IS SPECIFIED BY
                                            ROUTINE
0463   14600  066665         JMP  MNE      NO USER LABEL
0464*
0465   14601  035721         STB  T3       B-REG ADDRESS OF ROUTINE
0466   14602  070742         SAR  16
0467   14603  031722         STA  T4
0468   14604  163747         JSM  GAS2,I   GET CHARACTERS FROM LABEL
0469   14605  010546         CPA  QUOTE    END QUOTE?
0470   14606  066611         JMP  *+3      YES
0471   14607  045722         ISZ  T4       NO, COUNT CHARACTERS
0472   14610  066604         JMP  *-4      CONTINUE CHARACTER COUNTING
0473   14611  021722         LDA  T4
0474   14612  025722         LDB  T4
0475   14613  004420         ADB  DM16
0476   14614  074112         SBM  *+2      IS COUNT GREATER THAN 16
0477   14615  020364         LDA  DP16     YES SET COUNT TO 16
0478   14616  003737         ADA  DASS     SET BIT15 AND SUBTRACT ONE FROM
                                            COUNT
0479   14617  026643         LDB  ADR1
0480   14620  105655         ADB  PG,I     B-REG ADDRESS OF MNEMONIC
                                            BUILDING ROUTINE
0481   14621  170402         RET
```

(238)

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0483*
0484  14622  031720  SFFN   STA  T2        SAVE RELATIVE KEYCODE
0485  14623  025715         LDB  DFUN      START DEFINE-A-KEY AREA ADDRESS
0486  14624  074517  MNEM1  LDA  B,I
0487  14625  010405         CPA  DM1       FOUND PROGRAM.END?
0488  14626  066630         JMP  *+2       YES
0489  14627  077670         RIB  *-3       NO, RECYCLE
0490*
0491  14630  015543         CPB  LIM1      IS IT START OF MAIN-LINE AREA?
0492  14631  065357         JMP  1357B     YES, FUNCTION NOT DEFINE
0493*
0494  14632  074070         SIB  *+1
0495  14633  074517         LDA  B,I       GET ID
0496  14634  052720         AND  IDM
0497  14635  004402         ADB  DP2       SET ADDRESS TO FIRST LINE OF THIS
                                            PROGRAM
0498  14636  011720         CPA  T2        MATCH?
0499  14637  170402         RET            YES
0500  14640  004405         ADB  DM1
0501  14641  066624         JMP  MNEM1     RECYCLE
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0503*
0504   14642 000651 ADR2   DEF *+7-ID     ENTRY POINT FOR ALL CHARACTERS BUT
                                           THE FIRST
0505   14643 000644 ADR1   DEF *+1-ID     ENTRY POINT TO GET FIRST CHARACTER
0506*
0507   14644 022642        LDA ADR2       FIRST CHARACTER ROUTINE
0508   14645 101655        ADA PG,I
0509   14646 025620        LDB NCODE
0510   14647 031620        STA NCODE      SET UP ENTRY FOR ALL REMAINING
                                           CHARACTER
0511   14650 035723        STB T5         SAVE OLD VALUE OF NCODE
0512*
0513   14651 025721        LDB T3         T3 HAS ADDRESS WHERE LABEL CAN FOUND
0514   14652 163747        JSM GAS2,I     GET CHARACTER
0515   14653 035721        STB T3         SAVE ADDRESS
0516   14654 031720        STA T2         SAVE CHARACTER
0517   14655 055722        DSZ T4         CHECK COUNT
0518   14656 170402        RET            STILL MORE CHARACTERS
0519   14657 021723        LDA T5         NO MORE CHARACTERS
0520   14660 031620        STA NCODE      RESTORE NCODE
0521   14661 170402        RET
0522*
0523   14662 070742 MNE1   SAR 16         A-REG=0
0524   14663 024204        LDB 204B       B-REG CODE FOR P
0525   14664 170402        RET
0526*
0527   14665 021720 MNE    LDA T2         STANDARD MNEMONICS FOR OPTION BLOCK
0528   14666 002674        ADA MNTAB
0529   14667 070016        EXA
0530   14670 020402        LDA DP2
0531   14671 074132        SBM *+2,C
0532   14672 020401        LDA DP3        A-REG HAS CHARACTER COUNT -1
0533   14673 170402        RET            B-REG PACKED CODE FOR ASCII
                                           CHARACTERS
0534*
0535   14674 026676 MNTAB  LDB TABL+1
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0537  14675  046162  TABL    OCT  046162     SCR
0538  14676  022270          OCT  022270     IEX
0539  14677  006614          OCT  006614     CLL
0540  14700  176300          OCT  176300     ->F
0541  14701  000000          BSS  1          CHECKSUM
0542  14702  114040          OCT  114040     FA
0543  14703  114100          OCT  114100     FB
0544  14704  114140          OCT  114140     FC
0545  14705  114200          OCT  114200     FD
0546  14706  114240          OCT  114240     FE
0547  14707  044264          OCT  044264     RET
0548  14710  010246          OCT  010246     DEF
0549  14711  022706  TTB0    LDA  *-3
0550  14712  000045          OCT  45
0551  14713  000105          OCT  105
0552  14714  000117          OCT  117
0553  14715  000053  CNFP1   DEC  43
0554  14716  000054  CNFP2   DEC  44
0555  14717  177755  DM19    DEC  -19
0556  14720  001777  IDM     OCT  1777
0557  14721  116040          OCT  116040     GA
0558  14722  116100          OCT  116100     GB
0559  14723  116140          OCT  116140     GC
0560  14724  116200          OCT  116200     GD
0561  14725  116240          OCT  116240     GE
0562  14726  116300          OCT  116300     GF
0563  14727  116340          OCT  116340     GG
0564  14730  116400          OCT  116400     GH
0565  14731  116440          OCT  116440     GI
0566  14732  116500          OCT  116500     GJ
0567  14733  120040          OCT  120040     HA
0568  14734  120100          OCT  120100     HB
0569  14735  120140          OCT  120140     HC
0570  14736  120200          OCT  120200     HD
0571  14737  120240          OCT  120240     HE
0572  14740  120300          OCT  120300     HF
0573  14741  120340          OCT  120340     HG
0574  14742  120400          OCT  120400     HH
0575  14743  120440          OCT  120440     HI
0576  14744  120500          OCT  120500     HJ
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0578*
0579*  TRANSFER LINKS
0580*
0581   14745 062765 TINK   JSM  FIND   GET ADDRESS OF THE BLOCK
0582   14746 070402        SAR  9      A = 8,10,12
0583   14747 003724        ADA  NSRW   NON-SHARED R/W IN BASIC SYSTEM HAS
0584   14750 070517        LDA  A,I    START OF R/W AREA FOR THIS BLOCK
0585*
0586   14751 027725        LDB  OBBF   BASE ADDRESS FOR FIRST GROUP OF LINK
                                         ADDRESSES
0587   14752 035716        STB  T0     1650 OCTAL
0588   14753 024413        LDB  DM7    COUNTER FOR 8 WORDS
0589   14754 062760        JSM  TI1    CALL ROUTINE TO SET UP ADDRESS
0590*
0591   14755 027726        LDB  OBBF2  BASE ADDRESSES FOR SECOND GROUP
0592   14756 035716        STB  T0     1724 OCTAL
0593   14757 024406        LDB  DM2    COUNTER FOR 3 WORDS
0594   14760 131716 TI1    STA  T0,I   SET UP LINK ADDRESSES
0595   14761 045716        ISZ  T0
0596   14762 070070        SIA  *+1
0597   14763 077670        RIB  *-3
0598   14764 170402        RET
0599*
0600*  FIND ADDRESS OF THIS BLOCK
0601*
0602   14765 026555 FIND   LDB  OBRW   BASE ADDRESS OF OPTION BLOCK TABLE
                                         1626B
0603   14766 074517        LDA  B,I    GET CONFIGURATION CODE
0604   14767 052720        AND  IDM
0605   14770 012000        CPA  ID     DO ID'S MATCH
0606   14771 066773        JMP  *+2    YES, FOUND CORRECT WORD
0607   14772 077630        RIB  *-4
0608   14773 074517        LDA  B,I    GET PAGE ADDRESS
0609   14774 052552        AND  PGM    A-REG HAS BASE ADDRESS FOR THIS
                                         OPTION BLOCK
0610   14775 170402        RET
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0612*
0613*  HUNT UP START OF SUBROUTINE
0614*
0615  14776 062622 HUNT   JSM  SFFN     FIND FUNCTION
0616  14777 075713        SBP  *-2      IF B-REG + FUNCTION EXISTS
0617  15000 063264        JSM  KBFG+1
0618  15001 071610        SZA  *-4
0619*
0620  15002 025542 NEWF   LDB  END$     ELSE SET UP NEW FUNCTION
0621  15003 004400        ADB  DP4      INCREASE PROGRAM AREA BY 4 WORDS
0622  15004 074117        LDA  B
0623  15005 070056        CMA
0624  15006 001712        ADA  RR       BASE POINTER OF DATA REGISTER
                                          STORAGE
0625  15007 070113        SAP  *+2
0626  15010 065375        JMP  E9A      NOT ENOUGH ROOM ERROR #9
0627*
0628  15011 035542        STB  END$      THE 4-WORDS WILL FIT ADJUST END$ TO
0629  15012 035716        STB  T0       NEW VALUE
0630  15013 004410        ADB  DM4
0631  15014 074517        LDA  B,I      GET WORD FROM PROGRAM AREA
0632  15015 131716        STA  T0,I     MOVE IT DOWN
0633  15016 055716        DSZ  T0       ADJUST POINTER
0634  15017 015543        CPB  LIM1     HAS MAINLINE BEEN COMPLETELY MOVED
0635  15020 067023        JMP  *+3      YES
0636  15021 004405        ADB  DM1      ADJUST POINTER
0637  15022 067014        JMP  *-6      RECYCLE
0638*
0639  15023 035716        STB  T0       MAINLINE MOVE COMPLETE
0640  15024 021546        LDA  CMA      IF CMA IS
0641  15025 000400        ADA  DP4      IN MAINLINE AREA
0642  15026 070137        LDB  A
0643  15027 070076        TCA
0644  15030 001543        ADA  LIM1
0645  15031 070113        SAP  *+2
0646  15032 035546        STB  CMA      UPDATE IT
0647*
0648  15033 025716        LDB  T0
0649  15034 004400        ADB  DP4
0650  15035 035543        STB  LIM1     SET UP NEW LIM1
0651*
0652  15036 004407        ADB  DM3      STORE I.D.
0653  15037 021720        LDA  T2
0654  15040 074557        STA  B,I
0655  15041 074070        SIB  *+1
0656  15042 020405        LDA  DM1
0657  15043 074557        STA  B,I      SET -1 BOUNDARY FOR BEGINNING OF
                                          PROGRAM
0658  15044 074070        SIB  *+1
0659  15045 020521        LDA  STEN     SET FIRST LINE TO BE END
0660  15046 074557        STA  B,I
0661  15047 170402        RET
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0663*
0664*  SET UP ABSENTEE LINKS
0665*
0666   15050  074442 ECDS    SBR  10        B-REG ON ENTRY HAS 10000B,12000B, OR
                                            14000B
0667   15051  006554         ADB  ECRW      BEFORE ADD B-REG 4,5,6. 1631B,1632B,1633B
0668   15052  121655         LDA  PG,I
0669   15053  042001         IOR  ID+1      SYNTAX LINK
0670   15054  074557         STA  B,I
0671   15055  004401         ADB  DP3
0672   15056  121655         LDA  PG,I
0673   15057  042002         IOR  ID+2      EXECUTION LINK
0674   15060  074557         STA  B,I
0675   15061  004401         ADB  DP3
0676   15062  121655         LDA  PG,I
0677   15063  042003         IOR  ID+3      MNEMONIC LINK
0678   15064  074557         STA  B,I
0679   15065  121652         LDA  NAB,I
0680   15066  000401         ADA  DP3       UPDATE NAB BY 3
0681   15067  131652         STA  NAB,I     FOR EACH ADDITIONAL OPTION BLOCK
0682   15070  170402         RET
```

USER-DEFINABLE FUNCTION BLOCK—"UDFUN"

```
0684*
0685*  GET INTERNAL CODE
0686*
0687  15071  021544 GETIC   LDA SKEY
0688  15072  031717         STA  T1
0689  15073  000377         ADA DP5
0690  15074  070517         LDA A,I      LOWEST 8 BITS SPECIFY WHICH CODE OF THE
0691  15075  031716         STA  T0      OPTION BLOCKS. LOWEST 4 BITS WHICH OF
0692  15076  050365         AND DP15     16 FOR THE BLOCK. HIGHER 4 BITS WHICH
                                           BLOCK;
0693  15077  000405         ADA DM1      10000B = 1, 12000B = 2, 14000B = 3
0694  15100  031720         STA  T2
0695  15101  021716         LDA T0
0696  15102  070142         SAR 4
0697  15103  050401         AND DP3
0698  15104  000405         ADA DM1
0699  15105  031721         STA  T3
0700  15106  070644         SAL 3
0701  15107  001721         ADA T3
0702  15110  101652         ADA NAB,I    A=0,1,...,25,26
0703  15111  003115         ADA TAB
0704  15112  070016         EXA
0705  15113  001720         ADA T2       A=-1,0,1,...,37,38
0706  15114  170402         RET
0707*
0708  15115  023116 TAB     LDA *+1      TABLE OF INTERNAL CODES
0709  15116  000000         DEC 0
0710  15117  000055 CNFP3   DEC 45
0711  15120  000220 DP144   DEC 144
0712  15121  000000         DEC 0
0713  15122  000024         DEC 20
0714  15123  000024         DEC 20
0715  15124  000000         DEC 0
0716  15125  000024         DEC 20
0717  15126  000024         DEC 20
0718  15127  146000 SCODE   OCT 146000
0719  15130  000000         DEC 0
0720  15131  000063 CNFO    DEC 51
0721  15132  000024         DEC 20
0722  15133  000000         DEC 0
0723  15134  000024         DEC 20
0724  15135  000024         DEC 20
0725  15136  000000         DEC 0
0726  15137  000036         DEC 30
0727  15140  000034 CNFT    DEC 28
0728  15141  000052 CNFD    DEC 42
0729  15142  000000         DEC 0
0730  15143  000024         DEC 20
0731  15144  000024         DEC 20
0732  15145  000000         DEC 0
0733  15146  000036         DEC 30
0734  15147  000036         DEC 30
0735  15150  000000         DEC 0
```

USER-DEFINEABLE FUNCTION BLOCK—"UDFUN"

```
0737*
0738*  CLEAN UP SUBROUTINE AREA
0739*
0740   15151 001152 ADND   DEF *+1-ID
0741*
0742   15152 025543        LDB LIM1    THIS ROUTINE IS ADDED TO THE END
                                         ROUTINE
0743   15153 015715        CPB DFUN    ARE THERE PROTECTED SUBPROGRAMS?
0744   15154 067221        JMP CEXT1   NO, EXIT
0745   15155 035721        STB T3      YES, SAVE POINTER TO BEGINNING OF
                                         MAINLINES
0746   15156 025715        LDB DFUN    POINTER TO BEGINNING OF PROTECTED
                                         AREA
0747   15157 074517        LDA B,I     FIND A -2
0748   15160 010406        CPA DM2
0749   15161 067166        JMP *+5     FOUND A -2
0750   15162 074070        SIB *+1     CONTINUE TO LOOK BY MOVING POINTER
0751   15163 015721 CADB   CPB T3      HAVE WE REACHED MAINLINE AREA?
0752   15164 067206        JMP CEXT    YES
0753   15165 067157        JMP *-6     NO, RECYCLE
0754   15166 035717        STB T1      FIND END OF PROGRAM TO BE DELETED.
                                         SAVE CURRENT SEA
0755   15167 035720        STB T2      ADDRESS
0756   15170 074517        LDA B,I
0757   15171 010405        CPA DM1     FOUND END OF OLD PROGRAM?
0758   15172 067174        JMP *+2     YES
0759   15173 077670        RIB *-3     NO, RECYCLE
0760   15174 074517        LDA B,I       MOVE FUNCTION AREA
0761   15175 131720        STA T2,I    COVER HOLE
0762   15176 015721        CPB T3      COMPLETE?
0763   15177 067202        JMP *+3     YES
0764   15200 045720        ISZ T2      NO, ADJUST POINTER
0765   15201 077570        RIB *-5     ADJUST FETCH POINTER; RECYCLE
0766   15202 025720        LDB T2      SET B-REG TO SEARCH FOR MORE
                                         CLEAN-UP
0767   15203 035721        STB T3      ADJUST BEGINNING OF MAINLINE
0768   15204 025717        LDB T1      ADJUST FETCH POINTER
0769   15205 067163        JMP CADB    RECYCLE
0770*
0771   15206 025543 CEXT   LDB LIM1    MAIN CLEAR-UP IS COMPLETE BUT
0772   15207 021721        LDA T3      POINTERS NEED ADJUSTING
0773   15210 031543        STA LIM1    MAINLINE POINTER UPDATED
0774   15211 074517        LDA B,I     MOVE MAINLINE TO CORRECT LOCATION
0775   15212 131721        STA T3,I
0776   15213 015542        CPB END$    CORRECTION COMPLETED
0777   15214 067217        JMP *+3     YES
0778   15215 045721        ISZ T3      NO, ADJUST STORE POINTER
0779   15216 077570        RIB *-5     ADJUST THE FETCH POINTER
0780   15217 021721        LDA T3
0781   15220 031542        STA END$    UPDATE THE END-OF-MAINLINE POINTER
0782   15221 062745 CEXT1  JSM TINK
0783   15222 125654        LDB EN,I
0784   15223 074737        JMP B,I     DAISY CHAIN THROUGH END LINKAGE
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0786*
0787*  INTERPRETER
0788*
0789   15224  062745  EXEC    JSM  TINK      SET UP THIS BLOCK R/W AREA
0790   15225  062076          JSM  COPL3     CALL ROUTINE TO CLASSIFY KEYS
0791   15226  067612          JMP  XF        FUNCTION KEY RETURN
0792   15227  025720          LDB  T2        OTHER KEY RETURN. LOAD INTERNAL CODE
0793   15230  007716          ADB  TTB1      USE TTB1 TO LOCATE
0794   15231  074016          EXB            EXECUTION ROUTINE
0795   15232  074016          EXB
0796*
0797   15233  145653  XIEX    ISZ  IXFLG,I   "IEX" KEY. SET FLAG THAT THIS
0798   15234  164346          JMP  XJ,I      EVENT HAS HAPPENED. RETURN TO BASIC
                                              INTERPRETER
0799*
0800   15235  131653  SIEX    STA  IXFLG,I   "SCR IEX . . ." STATEMENT. CLEAR FLAG
0801   15236  067241          JMP  XSCR      CONTINUE SCANNING THE STATEMENT
0802*
0803   15237  027727  E24     LDB  DP4       ERROR #24. SCR STATEMENT CANNOT BE
0804   15240  167746          JMP  EXER,I    EXECUTED FROM PROGRAM
0805*
0806   15241  160552  XSCR    JSM  SXJ,I     BEGINNING OF A SCRATCH STATEMENT
0807   15242  010451          CPA  CNFC
0808   15243  067241          JMP  *-2       IGNORE ","
0809   15244  010356          CPA  CNFN
0810   15245  057261          JMP  SCB       END-OF-SCR STATEMENT. DO HOUSE
                                              CLEANING
0811   15246  063071          JSM  GETIC     GET INTERNAL CODE FOR CODE FOUND IN
                                              SCRATCH
0812   15247  031720          STA  T2        STATEMENT — SAVE IT
0813   15250  071250          SZA  SIEX      IS IT IEX? IF A-REG = 0 YES
0814   15251  063264          JSM  KBFG+1    KEYBOARD OPERATION OR PROGRAM?
0815   15252  071250  XSCR2   SZA  E24       IF A-REG = 0 PROGRAM, THEN ERROR #24
0816   15253  021720          LDA  T2
0817   15254  062776          JSM  HUNT      FIND SUBPROGRAM
0818   15255  004407          ADB  DM3       B-REG HAS STARTING ADDRESS OF
                                              PROTECTED PROGRAM
0819   15256  022553          LDA  ADM2      SET B-ADDRESS BACK 3 WORDS, BEGINNING
                                              ADDRESS FOR -2
0820   15257  170004          XFR            TRANSFER FOUR WORDS INTO PROTECTED
                                              PROGRAM
0821   15260  067241          JMP  XSCR      CONTINUE SCANNER OF THE STATEMENT
0822*
0823   15261  161710  SCB     JSM  .END2,I   STATEMENT COMPLETELY SCANNED CALL
0824   15262  164346          JMP  XJ,I      END CODE PROCESSING FOR HOUSE
                                              CLEANING
0825*
0826   15263  061253  KBFG    JSM  XNSTP     CHECK FOR KEYBOARD OPERATION OR
                                              PROGRAM
0827   15264  021563          LDA  TSW       ROUTINE. TSW = 4 IF PROGRAM
0828   15265  000410          ADA  DM4       A-REG IS ZERO IF PROGRAM MODE
0829   15266  070110          SZA  *+2       NONZERO IF KEYBOARD
0830   15267  021546          LDA  CMA
0831   15270  170402          RET
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0833*
0834  15271 201705 XSEOL  LDA  .WMOD   SPECIAL EOL KEYCODE. SET UP STACK FOR
0835  15272 050512        AND  512B    CLL STMTS.
0836  15273 073350        RZA  SCB+1   IF ERROR IN LINE ABORT ROUTINE
0837  15274 160552        JSM  SXJ,I   JUMP OVER NORMAL EOL
0838  15275 070742        SAR  16
0839  15276 031702        STA  GTS     CLEAR RUN-TIME GTO-GSB FLAG
0840  15277 121605        LDA  IMA,I   SET UP PROPER RETURN ADDRESS
0841  15300 070133        SAP  *+2,C   *
0842  15301 070070        SIA  *+1     *
0843  15302 055556        DSZ  AP1     UPDATE STACK POINTER
0844  15303 025556        LDB  AP1
0845  15304 074557        STA  B,I     STORE R.A. IN STACK
0846  15305 063263        JSM  KBFG    WHAT MODE? KEYBOARD OR PROGRAM
0847  15306 004405 XSE1   ADB  DM1
0848  15307 074557        STA  B,I     SAVE INFO ABOUT KEYBD OR PROG.
0849  15310 020400        LDA  DP4
0850  15311 031563        STA  TSW     SET TSW=4 PROGRAM MODE
0851  15312 004405        ADB  DM1
0852  15313 021614        LDA  TPP
0853  15314 074557        STA  B,I     SAVE RESET POINTER FOR TEMPORARY
                                       STORAGE
0854  15315 021551        LDA  TP
0855  15316 031614        STA  TPP     UPDATE TEMPORARY (RESET) STORAGE
                                       POINTER
0856  15317 004405        ADB  DM1
0857  15320 121650        LDA  NUMP,I
0858  15321 074557        STA  B,I     SAVE CURRENT # OF PARAMETER
0859  15322 063340        JSM  SETL    CHECK TO SEE IF INITIALIZATION IS
                                       NECESSARY
0860  15323 021556        LDA  AP1
0861  15324 070056        CMA
0862  15325 001557        ADA  AP2
0863  15326 131650        STA  NUMP,I  SET UP # PARAMETERS FOR THIS LEVEL
0864  15327 004405        ADB  DM1
0865  15330 121725        LDA  L,I     GET BASE ADDRESS FOR
0866  15331 074557        STA  B,I     DATA STORAGE FOR LOCAL VARIABLES
                                       FOR THIS LEVEL
0867  15332 121657        LDA  LVADD,I HIGH WATER MARK FOR THIS LEVEL FOR
                                       LOCAL VARIABLE
0868  15333 131725        STA  L,I     MAKE IT NEW BASE ADDRESS FOR NEXT
                                       LEVEL
0869  15334 004405        ADB  DM1
0870  15335 021557        LDA  AP2     GET POINTER WHICH SEGMENTS RUN TIME
                                       STACK
0871  15336 072053        SAP  *+1,S   SET SPECIAL FLAG TO INDICATE
                                       PARAMETERS PASSED
0872  15337 167755        JMP  XGSS1,I FINISH STACK ROUTINE WITH BASIC
                                       SYSTEM ROUTINE
0873*
0874  15340 121557 SETL   LDA  AP2,I   ROUTINE USED TO CHECK THE NEED
0875  15341 000403        ADA  DP1     TO INITIALIZE LVADD AND L
0876  15342 072350        RZA  *+7     WHICH ARE THE HIGH WATER MARK AND
                                       BASE
0877  15343 131650        STA  NUMP,I  ADDRESS FOR LOCAL VARIABLE TO
0878  15344 021542        LDA  END     SUBPROGRAMS WHEN P( ) IS USED
0879  15345 001615        ADA  BSTK
0880  15346 000407        ADA  DM3
0881  15347 131657        STA  LVADD,I
0882  15350 131725        STA  L,I
0883  15351 170402 RET    RET
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0885*
0886  15352  160552 XEND   JSM  SXJ,I     SPECIAL "END" KEY, ACTS AS RETURN FOR
                                          PROTECTED PROG
0887*
0888  15353  125557 XRET   LDB  AP2,I     SPECIAL "RET" KEY.
0889  15354  063264        JSM  KBFG+1    KEYBOARD OR PROGRAM MODE
0890  15355  070150        SZA  *+3       PROGRAMS IF ZERO
0891  15356  025557        LDB  AP2       KEYBOARD ; SET B-REG TO AP2
0892  15357  067412        JMP  XRET1
0893  15360  014405        CPB  DM1       IF B-REG = -1 NO RETURN ADDRESS IN THE
0894  15361  067425        JMP  E7        STACK; THEREFOR ERROR #7
0895  15362  074753        SBP  XRET2     IF BIT15=0 SIMPLE RETURN
0896  15363  045557        ISZ  AP2       BIT15=1 RETURN FOR FUNCTION OR CALLS
0897  15364  121725        LDA  L,I
0898  15365  131657        STA  LVADD,I   SET HIGH WATER MARK TO LOWER LEVEL
0899  15366  121557        LDA  AP2,I
0900  15367  131725        STA  L,I       GET BASE ADDRESS FOR LOWER LEVEL
0901  15370  045557        ISZ  AP2
0902  15371  121557        LDA  AP2,I
0903  15372  131650        STA  NUMP,I    GET PARAMETER COUNT FOR LOWER LEVEL
0904  15373  045557        ISZ  AP2
0905  15374  121557        LDA  AP2,I
0906  15375  031614        STA  TPP       GET LOWER LEVEL POINTER TO SYSTEM
                                          TEMPORARY
0907  15376  045557        ISZ  AP2
0908  15377  121557        LDA  AP2,I
0909  15400  072510        RZA  XRET1     KEYBOARD OR PROGRAM MODE FLAG
0910  15401  045557 XRET2  ISZ  AP2       PROGRAM MODE
0911  15402  121557        LDA  AP2,I     GET RETURN ADDRESS
0912  15403  010406        CPA  DM2
0913  15404  067417        JMP  XRET4     IF -2 SPECIAL FLAG FOR IEX
0914  15405  031604 XRET3  STA  NMA       SET RETURN ADDRESS IN PROGRAM
                                          POINTER
0915  15406  031546        STA  CMA
0916  15407  074073        SBP  *+1,C
0917  15410  035557        STB  AP2       RESET TO LOWER LEVEL IN RUN TIME STACK
0918  15411  164346        JMP  XJ,I      RETURN TO BASIC SYSTEM INTERPRETER
0919  15412  061250 XRET1  JSM  XSTOP     KEYBOARD MODE SET STOP BIT
0920  15413  070742        SAR  16
0921  15414  031624        STA  DISEN     CLEAR ENTER FLAG
0922  15415  021543        LDA  LIM1      SET R.A. FOR BEGINNING OF MAINLINE
0923  15416  073370        RIA  XRET3     FIRST RETURN ROUTINE
0924  15417  061250 XRET4  JSM  XSTOP     IEX MODE SET STOP BIT
0925  15420  070742        SAR  16
0926  15421  031624        STA  DISEN     CLEAR ENTER INHIBIT FLAG
0927  15422  045557        ISZ  AP2
0928  15423  121557        LDA  AP2,I     CONTINUE FOR R.A. ADDRESS
0929  15424  067406        JMP  XRET3+1
0930*
0931  15425  024375 E7     LDB  DP7       ERROR #7
0932  15426  167746        JMP  EXER,I    RET HAS NO MATCHING GSB
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0934*
0935   15427  001430  ADGT    DEF  *+1-ID
0936   15430  160552          JSM  SXJ,I     ROUTINE USED TO RETURN AN
0937   15431  062745          JSM  TINK      ADDRESS FOR PROTECTED SUBPROGRAMS
0938   15432  063071  ADGT1   JSM  GETIC     WHEN F*, G*, OR H* IS USED
0939   15433  025715          LDB  DFUN      WITH GSB, GTO, OR CLL
0940   15434  074070          SIB  *+1
0941   15435  070112          SAM  *+2
0942   15436  062776          JSM  HUNT      B-REG HAS ADDRESS OF ROUTINE
0943   15437  167756  ADGT2   JMP  XG2,I     RETURN TO BASIC SYSTEM PROCESSING OF
                                              GSB OR GTO

0944*
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
0946*
0947  15440 160552 XDEF   JSM  SXJ,I     "DEF STMT". GET NEXT KEYCODE
0948  15441 062745        JSM  TINK    SETUP R/W AREA FOR THIS BLOCK
0949  15442 063071        JSM  GETIC
0950  15443 062776        JSM  HUNT    LOCATE OR SETUP PROTECTED PROGRAM
0951  15444 004406        ADB  DM2
0952  15445 074517        LDA  B,I
0953  15446 072052        SAM  *+1,S.  SET B15 IN ID TO INDICATE THE
0954  15447 074557        STA  B,I     PROGRAM WAS DEFINED WITH DEF
0955  15450 004402        ADB  DP2     AS APPOSED TO GTO.
0956  15451 035604        STB  NMA
0957  15452 035546        STB  CMA     SET UP STORAGE POINTER.
0958  15453 164346        JMP  XJ,I    RETURN TO BASIC SYSTEM INTERPRETER
0959*
0960  15454 055777 CIEX   DSZ  RSTAK   ROUTINE TO HANDLE IEX ROUTINES.
                                       ADJUST RETURN STACK
0961  15455 063340        JSM  SETL    INITIALIZE LVADD AND L IF NECESSARY
0962  15456 021563        LDA  TSW
0963  15457 000411        ADA  DM5
0964  15460 072150        RZA  *+3     TSW=5? NO SKIP 3 INSTRUCTIONS
0965  15461 045624        ISZ  DISEN   YES,SYSTEM IS IN ENTER MODE. SET
                                       DISABLE ENTER FLAG
0966  15462 067464        JMP  *+2
0967  15463 160315        JSM  INTRP,I SETUP RUN-TIME STACK
0968  15464 021557        LDA  AP2
0969  15465 031556        STA  AP1     RESET AP1
0970  15466 061250        JSM  XSTOP   SET STOP BIT
0971  15467 074117        LDA  B
0972  15470 024414        LDA  AISP
0973  15471 035605        STB  IMA  SET ADDRESS OF INTERPRETER TO ISP
0974  15472 163754        JSM  XGSS2,I SAVE ADDRESS OF SUBROUTINE
0975  15473 163757        JSM  RKBU1,I EXECUTE COMPILER OUTPUT AREA
0976  15474 067476        JMP  *+2     NORMAL
0977  15475 170402        RET          ERROR RETURN
0978  15476 121557        LDA  AP2,I
0979  15477 031557        STA  AP2
0980  15500 000405        ADA  DM1
0981  15501 031556        STA  AP1
0982  15502 070537        LDB  A,I     FETCH ADDRESS OF SUBPROGRAM
0983  15503 035570        STB  ISP     SAVE IN ISP
0984  15504 074073        SBP  *+1,C
0985  15505 021546        LDA  CMA
0986  15506 131556        STA  AP1,I
0987  15507 035546        STB  CMA     SETUP RETURN ADDRESS. IEX PROGRAMS
                                       ARE SUBPROGRAMS
0988  15510 061250        JSM  XSTOP   SET STOP SO XGSS2 ROUTINE CAN BE USED
0989  15511 020406        LDA  DM2     SPECIAL FLAG FOR RETURN ROUTINE
0990  15512 163754        JSM  XGSS2,I CALL ROUTINE TO SAVE FLAG IN STACK
0991  15513 061253        JSM  XNSTP   CLEAR STOP BIT
0992  15514 021624        LDA  DISEN
0993  15515 072210        RZA  *+4     IS SYSTEM IN ENTER MODE?
0994  15516 163761        JSM  RPBUF,I NO, CALL INTERPRETER
0995  15517 167765        JMP  RU71,I  RETURN TO SYSTEM SAME AS RUN
                                       PROGRAM
0996  15520 167762        JMP  EXERR,I ERROR RETURN
0997*
0998  15521 163761        JSM  RPBUF,I YES, ENTER MODE?
0999  15522 167764        JMP  STSW5,I NORMAL RETURN SET TSW=5
1000  15523 167763        JMP  TSW5,I  ERROR RETURN
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
1002*
1003  15524  063340  XTAU   JSM  SETL   BEGINNING OF FUNCTION CALLED CODE.
                               INITIALIZE LVADD
1004  15525  021551         LDA  TP
1005  15526  000400         ADA  DP4
1006  15527  031551         STA  TP     SAVE A TEMPORARY RESULT LOCATION
                               FOR FUNCTION VALUE
1007  15530  072053         SAP  *+1,S
1008  15531  025556         LDB  AP1
1009  15532  004405         ADB  DM1
1010  15533  074557         STA  B,I    SET THIS ADDRESS IN CORRECT POSITION
                               OF STACK
1011  15534  004405         ADB  DM1
1012  15535  021563         LDA  TSW
1013  15536  074557         STA  B,I    SAVE CURRENT VALUE OF TSW IN STACK
1014  15537  004405         ADB  DM1
1015  15540  063263         JSM  KBFG   KEYBOARD OR PROGRAM MODE
1016  15541  070110         SZA  *+2    IF ZERO PROGRAM MODE
1017  15542  045624         ISZ  DISEN  KEYBOARD DISALLOW ENTER MODE
1018  15543  074557         STA  B,I    SAVE INFO ABOUT KEYBD OR PRG MODE
1019  15544  020400         LDA  DP4
1020  15545  031563         STA  TSW    SET TSW=4
1021  15546  004405         ADB  DM1
1022  15547  021604         LDA  NMA    SAVE NMA. MAYBE GSB OR GTO STAT.
                               PRECEEDED FUNCTION
1023  15550  074557         STA  B,I
1024  15551  070742         SAR  16     CLEAR NMA
1025  15552  031604         STA  NMA
1026  15553  004405         ADB  DM1
1027  15554  021557         LDA  AP2
1028  15555  035556         STB  AP1
1029  15556  035557         STB  AP2    SETUP NEW LEVEL WITHIN RUN-TIME STACK
1030  15557  131557         STA  AP2,I  RETURN TO INTERPRETER
1031  15560  164346         JMP  XJ,I
1032*
1033  15561  131605  XFN2   STA  IMA,I
1034  15562  160552         JSM  SXJ,I  GET FUNCTION KEYCODE
1035  15563  063071         JSM  GETIC  GET INTERNAL CODE
1036  15564  062776         JSM  HUNT
1037  15565  074113         SBP  *+2    WAS SUBPROGRAM DEFINED
1038  15566  067437         JMP  ADGT2  NO,SETUP FUNCTION
1039  15567  067604         JMP  XFN1   YES
1040*
1041  15570  160552  XFUNC  JSM  SXJ,I  END OF PARAMETER LIST. SECOND SPECIAL
                               FUNCTION
1042  15571  121605         LDA  IMA,I  CALL CODE. SEARCH FOR ADDRESS OF
                               SUBPROGRAM
1043  15572  074073         SBP  *+1,C
1044  15573  035722         STB  T4
1045  15574  070070         SIA  *+1
1046  15575  055556         DSZ  AP1
1047  15576  131556         STA  AP1,I  A-REG HAS CORRECT ADDRESS FOR RETURN
                               TO ARITH STATE
1048  15577  025567         LDB  MPH    CHECK FAST TRANSFER FLAG
1049  15600  077050         RZB  XFN2   IF NON-ZERO FAST TRANSFER NOT ALLOWED
1050  15601  125722         LDB  T4,I   FAST TRANSFER ALLOWED. GET 16-BIT
                               ADDRESS
1051  15602  076110         RZB  *+2    IF ZERO,FIRST TIME FUNCTION CALLED
1052  15603  067561         JMP  XFN2   SEARCH FOR SUBPROGRAM
1053  15604  135722  XFN1   STB  T4,I   B-REG HAS ADDRESS OF THE SUBPROGRAM
1054  15605  135635         STB  IMA,I  SAVE IT IN USER'S PROGRAM
1055  15606  035546         STB  CMA    UPDATE CMA
1056  15607  025556         LDB  AP1
1057  15610  070742         SAR  16
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
1058  15611  067306          JMP  XSE1     CONTINUE BUILDING RUN-TIME STACK
1059*
1060  15612  025557  XF      LDB  AP2      RETURN TO ARITHMETIC STATEMENT.
1061  15613  074517          LDA  B,I        FIRST CODE IN FUCTION KEY CODE.
1062  15614  031557          STA  AP2        UNSTACK RUN-TINE STACK BUILT
                                             BY TAU. RESET AP2
1063  15615  074070          SIB  *+1
1064  15616  074517          LDA  B,I
1065  15617  031604          STA  NMA      RESET NMA
1066  15620  074070          SIB  *+1
1067  15621  074517          LDA  B,I
1068  15622  072510          RZA  XF2      IF NON-ZERO FUNCTION CALLED
                                             FROM KEYBOARD
1069  15623  074070  XF1     SIB  *+1      ELSE FUNCTION WAS IN PROGRAM
1070  15624  074517          LDA  B,I
1071  15625  031563          STA  TSW      RESET TSW
1072  15626  074070          SIB  *+1
1073  15627  074517          LDA  B,I
1074  15630  035556          STB  AP1      RESET AP1
1075  15631  070073          SAP  *+1,C
1076  15632  031551          STA  TP       RESET TP
1077  15633  164346          JMP  XJ,I     RETURN TO MAINLINE INTERPRETER
1078*
1079  15634  031546  XF2     STA  CMA      FUNCTION WAS CALLED FROM
                                             KEYBOARD. READJUST CMA
1080  15635  061250          JSM  XSTOP    SET STOP BIT
1081  15636  070742          SAR  16
1082  15637  031624          STA  DISEN    CLEAR ENTER MODE FLAG
1083  15640  067623          JMP  XF1
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
1085*
1086  15641  025557  XGAZ  LDB  AP2      ->F ROUTINE
1087  15642  074517        LDA  B,I
1088  15643  010405        CPA  DM1      IF AP2 IS POINTING TO A -1
1089  15644  067703        JMP  E2       THEN ERROR #2. ->F KEY APPEARS IN A
                                          PROGRAM NOT C
1090  15645  070172        SAM  *+3,C    WITH A FUNCTION. LOOK FOR SPECIAL
                                          FLAG
1091  15646  070137        LDB  A        LOOK FOR ANOTHER AP2
1092  15647  067642        JMP  *-5
1093  15650  000400        ADA  DP4
1094  15651  070537        LDB  A,I      GET POINTER TO CORRECT TEMPORARY
1095  15652  074073        SBP  *+1,C    CLEAR TEMPORARY FLAG
1096  15653  035550        STB  .STPT    SETUP .STPT WHICH IS POINTER FOR
                                          NUMERIC DISPLAY
1097  15654  167745        JMP  XGAZ1,I  B—REG HAS THE "TO" ADDRESS. CALL
                                          REMAINDER OF GAZI
1098*
1099  15655  061321  XPAR  JSM  FIXPT    "P" ROUTINE CONVERT SUBSCRIPT TO
                                          BINARY
1100  15656  074153        SBP  *+3      IF — ERROR #5
1101  15657  024377  E5    LDB  DP5
1102  15660  167746        JMP  EXER,I
1103  15661  075710        SZB  *-2      IF ZERO ERROR #5
1104  15662  074076        TCB           MAKE IT NEGATIVE
1105  15663  035717        STB  T1       SAVE SUBSCRIPT
1106  15664  025557        LDB  AP2      FIND STACK AREA WHERE PARAMETER
                                          ADDRESS EXISTS
1107  15665  074517        LDA  B,I      *
1108  15666  010405        CPA  DM       *
1109  15667  067703        JMP  E2       *
1110  15670  070172        SAM  *+3,C    *
1111  15671  070137        LDB  A        *
1112  15672  067665        JMP  *-5      *
1113  15673  025717        LDB  T1
1114  15674  105650        ADB  NUMP,I
1115  15675  074412        SBM  XLOV     IF + SUBSCRIPT REFERS TO PARAMETER
                                          IN LIST
1116  15676  001717        ADA  T1
1117  15677  070517        LDA  A,I      GET PARAMETER ADDRESS
1118  15700  070073        SAP  *+1,C    CLEAR FLAG BIT 15
1119  15701  131556  XP1   STA  AP1,I    STORE OPERAND ADDRESS IN CORRECT
                                          POSITION OF STACK
1120  15702  164346        JMP  XJ,I     RETURN TO INTERPRETER
1121*
1122  15703  024402  E2    LDB  DP2
1123  15704  167746        JMP  EXER,I
1124*
1125  15705  074076  XLOV  TCB           SUBSCRIPT IS OUTSIDE THE RANDE OF
                                          THE PARAMETER LIS
1126  15706  074704        SBL  2
1127  15707  105725        ADB  L,I      COMPUTE CORRECT ADDRESS FROM
                                          BASE ADDRESS IN L
1128  15710  074117        LDA  B
1129  15711  074076        TCB
1130  15712  105657        ADB  LVADD,I
1131  15713  075313        SBP  XP1
1132  15714  131657        STA  LVADD,I  ADJUST HIGH-WATER MARK IF
                                          NECESSARY
1133  15715  067701        JMP  XP1
```

(254)

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
1135*
1136*  INTERPRETER TABLE
1137*
1138   15716  027720  TTB1    LDB  *+2
1139   15717  067241          JMP  XSCR
1140   15720  067233          JMP  XIEX
1141   15721  167752          JMP  XGTO,I  XCALL
1142   15722  067641          JMP  XGAZ
1143   15723  067655          JMP  XPAR
1144   15724  001632  NSRW    DEF  1632B
1145   15725  001650  OBBF    DEF  1650B
1146   15726  001724  OBBF2   DEF  1724B
1147   15727  000030  DP24    DEC  24
1148   15730  000035  GSBK    OCT  35
1149   15731  067353          JMP  XRET
1150   15732  067440          JMP  XDEF
1151   15733  067352          JMP  XEND
1152   15734  067271          JMP  XSEOL
1153   15735  067570          JMP  XFUNC
1154   15736  067524          JMP  XTAU
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
1156*
1157   00431          OB1     EQU  431B
1158   00432          OB2     EQU  432B
1159   00433          OB3     EQU  433B
1160   00005          IGN     EQU  5
1161*
1162   00012          CLOD    EQU  12B
1163   00144          GAZ     EQU  144B
1164   00077          REG     EQU  77B
1165   00141          SQRT    EQU  141B
1166   00254          SKCD    EQU  254B
1167   00314          BCPLR   EQU  314B
1168   00315          INTRP   EQU  315B
1169   00316          XEOL    EQU  316B
1170   00323          .SMON   EQU  323B
1171   00333          ASTEN   EQU  333B
1172   00345          RH      EQU  345B
1173   00351          O77     EQU  351B
1174   00356          DP47    EQU  356B
1175   00362          XRH     EQU  362B
1176   00364          DP16    EQU  364B
1177   00365          DP15    EQU  365B
1178   00366          DP14    EQU  366B
1179   00367          DP13    EQU  367B
1180   00370          DP12    EQU  370B
1181   00371          DP11    EQU  371B
1182   00372          DP10    EQU  372B
1183   00373          DP9     EQU  373B
1184   00374          DP8     EQU  374B
1185   00375          DP7     EQU  375B
1186   00376          DP6     EQU  376B
1187   00377          DP5     EQU  377B
1188   00400          DP4     EQU  400B
1189   00401          DP3     EQU  401B
1190   00402          DP2     EQU  402B
1191   00403          DP1     EQU  403B
1192   00405          DM1     EQU  405B
1193   00406          DM2     EQU  406B
1194   00407          DM3     EQU  407B
1195   00410          DM4     EQU  410B
1196   00411          DM5     EQU  411B
1197   00413          DM7     EQU  413B
1198   00414          AISP    EQU  414B
1199   00420          DM16    EQU  420B
1200   00446          ALCOM   EQU  446B
1201   00451          DP20    EQU  451B
1202   00517          BQT     EQU  517B
1203   00500          APMA    EQU  500B
1204   00507          NULL    EQU  507B
1205   00521          STEN    EQU  521B
1206   00524          XRI     EQU  524B
1207   00533          RET1    EQU  533B
1208   00537          COEOL   EQU  537B
1209   00546          QUOTE   EQU  546B
1210   00551          OPTN    EQU  551B
1211*
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
1212  00451   CNFC   EQU  DP20
1213  00356   CNFN   EQU  DP47
1214  00355   CNFR   EQU  355B
1215  00454   CNFY   EQU  454B
1216  00452   CNFQ   EQU  452B
1217  00437   EOL    EQU  437B
1218  00503   CNFL   EQU  503B
1219  00457   CNFE   EQU  457B
1220  00355   CNFB   EQU  355B
1221*
1222  01542   END$   EQU  1542B
1223  01543   LIMI1  EQU  1543B
1224  01544   SKEY   EQU  1544B
1225  01545   PSP    EQU  1545B
1226  01546   CMA    EQU  1546B
1227  01550   .STPT  EQU  1550B
1228  01551   TP     EQU  1551B
1229  01556   AP1    EQU  1556B
1230  01557   AP2    EQU  1557B
1231  01560   OSW    EQU  1560B
1232  01561   CODE   EQU  1561B
1233  01540   CN     EQU  1540B
1234  01541   OP     EQU  1541B
1235  01563   TSW    EQU  1563B
1236  01567   MPH    EQU  1567B
1237  01570   ISP    EQU  1570B
1238  01573   TECS   EQU  1573B
1239*
1240  01600   PMA    EQU  1600B
1241  01601   STAT   EQU  1601B
1242  01602   G      EQU  1602B
1243  01604   NMA    EQU  1604B
1244  01605   IMA    EQU  1605B
1245  01606   PS     EQU  1606B
1246  01620   NCODE  EQU  1620B
1247  01614   TPP    EQU  1614B
1248  01615   BSTK   EQU  1615B
1249  01616   ERASE  EQU  1616B
1250  01623   GOTF   EQU  1623B   GTO LINKAGE
1251  01624   DISEN  EQU  1624B
1252*
1253  01650   NUMP   EQU  1650B
1254  01651   .G     EQU  1651B
1255  01652   NAB    EQU  1652B   CONFIGURATION
1256  01653   IXFLG  EQU  1653B
1257  01654   EN     EQU  1654B   END LINKAGE
1258  01655   PG     EQU  1655B   PAGE IDENTIFICATION
1259  01656   SOSW   EQU  1656B
1260  01657   LVADD  EQU  1657B
1261  01666   POPR   EQU  1666B
1262*
1263  01701   .WKC   EQU  1701B
1264  01705   .WMOD  EQU  1705B
1265  01710   .END2  EQU  1710B
1266  01712   RR     EQU  1712B
1267  01715   DFUN   EQU  1715B
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
1268*
1269   01716          T0       EQU  1716B
1270   01717          T1       EQU  1717B
1271   01720          T2       EQU  1720B
1272   01721          T3       EQU  1721B
1273   01722          T4       EQU  1722B
1274   01723          T5       EQU  1723B
1275   01724          EOLFG    EQU  1724B
1276   01725          L        EQU  1725B
1277   01726          IEXC     EQU  1726B
1278   01744          AR1      EQU  1744B
1279   01702          GTS      EQU  1702B
1280   01777          RSTAK    EQU  1777B
1281*
1282   01250          XSTOP    EQU  1250B
1283   01253          XNSTP    EQU  1253B
1284   01265          XRR      EQU  1265B
1285   01274          DISIS    EQU  1274B
1286   01304          CP1      EQU  1304B
1287   01305          CP2      EQU  1305B
1288   01317          E1       EQU  1317B
1289   01321          FIXPT    EQU  1321B
1290   01351          BAD      EQU  1351B
1291   01362          EXIT     EQU  1362B
1292   01375          E9A      EQU  1375B
1293*
1294   15737  077777  DASS     OCT  77777
1295*
1296*  EDIT-DEPENDENT SYSTEM LINKS
1297*
1298   15740  004126  FCN      DEF  4126B
1299   15741  004315  CC       DEF  4315B
1300   15742  004317  CC1      DEF  4317B
1301   15743  004350  STSK     DEF  4350B
1302   15744  004422  CHKR     DEF  4422B
1303   15745  004672  XGAZ1    DEF  4672B
1304   15746  004705  EXER     DEF  4705B
1305   15747  005401  GAS2     DEF  5401B
1306   15750  005334  GQ2      DEF  5334B
1307   15751  005526  GTOP     DEF  5526B
1308   15752  005213  XGTO     DEF  5213B
1309   15753  005342  GQ4      DEF  5342B
1310   15754  005172  XGSS2    DEF  5172B
1311   15755  005177  XGSS1    DEF  5177B
1312   15756  005276  XG2      DEF  5276B
1313   07646          RKBUF    EQU  7646B
1314   15757  007647  RKBU1    DEF  RKBUF+1
1315   15760  007023  SKCD1    DEF  7023B
1316   15761  007656  RPBUF    DEF  7656B
1317   06247          A.11     EQU  6247B
1318   15762  006251  EXERR    DEF  A.11+2
1319   15763  006257  TSW5     DEF  6257B
1320   15764  006263  STSW5    DEF  6263B
1321   15765  006646  RU71     DEF  6646B
1322   15766  007746  RG4      DEF  7746B
1323   00326          RECY     EQU  326B
```

USER-DEFINEABLE FUNCTION BLOCK — "UDFUN"

```
1324   00346          XJ      EQU  346B
1325   00552          SXJ     EQU  552B
1326   15767  016067  LOOP    DEF  16067B
1327   15770  016200  C       DEF  16200B
1328   15771  016276  STORE   DEF  16276B
1329   15772  016235  INSTK   DEF  16235B
1330   15773  016246  CHECO   DEF  16246B
1331   15774  016323  NEXTS   DEF  16323B
1332*
1333   15775  000000          NOP            RESERVED FOR CHECKSUM
1334*
1335                          END
*   NO ERRORS*
```

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 5K words of memory, organized into 4K × 16 ROM, and 1024 × 16 RWM. The 16 bit RWM words are divided into user registers and processor words. The largest machine contains 7K words of ROM and 2K words of RWM.

Read/Write Memory.

As shown in Figure 11, memory is made up of 1024 × 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 μs at least very 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 μa 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 Figure 11, ROM chips are 4096 bit, n-channel MOS arranged 512 × 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.

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 × 64 bit storage array. For example, let I_0—I_5=0 and I_6—I_7 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 or 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_8 or its complement (I_8). 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 × 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 $512_{10} \times 8_{10}$ ROM chips by organizing the ROM chips into 64 × 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 100_8. For example, the column number of address $000 = 100_8 - 00_8 - 100 = 64_{10}$ and the column number of address $777 = 100_8 - 77_8 = 1$.

M-Register.

As shown in Figures 11 and 12A—D, 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 TTT 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 output and 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 the 5 most significant bits. The 5 least significant address bits are bussed to the control card where they are used in part of the refresh circuitry.

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 11 and 13A—D, control uses these pulses and instructions to generate the clocks required by the RWM chips. A synchronous 4 bit counter (SN74193) is used to count clock pulses and the 4 outputs are decoded by a 1 and 16 decoder (SN74154) to generate J and K input to flip-flops. 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 card. 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, $\overline{CCT}$, is high. $\overline{CCT}$ 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 $\overline{INH}$. $\overline{INH}$ halts the machine and the refresh cycle begins.

The same counter used, for a memory cycle is used during refresh to again generate the necessary clocks (Pre-charge and chip select). When the counter returns to state 0 and REF is present, a second counter is advanced one count. This second 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 card is to provide for extended memory capability. The control card 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 card 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 card 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 11 and 14A—D, 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 $\overline{TTT}$ instruction, and simultaneously recirculated into 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 nane gates. All 16 bits are read from either RWM or ROM simultaneously. Each bit to be written into RWM has its own discrete buffer stage that translates T^2L logic levels into 16v logic levels used by the RWM.

MEMORY SYSTEM MNEMONIC TABLE.
SIGNALS GENERATED OUTSIDE MEMORY I/O CONNECTOR.

CCT — Control clock-not, the inverted envelop of SCK.

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 $\overline{S}$-BUS.

ITS — I/O to $\overline{S}$-BUS, the signal which gates IOD to $\overline{S}$-BUS.

SCO & — Coded signals which generate UTS — Units to $\overline{S}$-BUS
ZTS — Zero „ „
MTS — M-Reg „ „
TTS — T-Reg „ „

$\overline{TTT}$ — T-BUS to T-Reg, OV = True.

T-BUS — Data on this bus acts as inputs to M & T registers.

$\overline{RDM}$ — Read memory, negative true. Lasts for 12 clock pulses.

$\overline{WTM}$ — Write memory, negative time. Lasts for 12 clock pulses.

$\overline{INH}$ — 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.

OTHER SIGNALS AT I/O CONNECTOR

Name	Source
T00	T-Register
T01	„ „
T02	„ „
T03	„ „

D-BUS — Data Bus — external data (extended memory data) enters machine via this bus.

$\overline{EDT}$ — External Data Transfer gates D-Bus data into machine 0^v = True.

$\overline{EMB}$ — 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.
600μa into 150Ω = Ω 1″
0 current = 0

$\overline{AOF}$ — Add-on function not .5^v signal if the add on R/W board (option Ø1) is not present in the machine. 0^v = True.

OTHER SIGNALS USED BY (RWM).

Name	Source
A00—A04	CONTROL
A05—A09	M-REG
CSB	M-REG
CSA	M-REG
RWI(XX)	T-REG
YBL	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.

Name	Source
ROD(XX)	ROM
TRI	CONTROL
TSC	CONTROL
$\overline{TMC}$	CONTROL
TPC	CONTROL
RWD(XX)	R/W MEM
RWE	M-REG
MAW	M-REG

$\overline{AOF}$ — Add on R/W Board (Option 01)

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 the two bits available at ROM output buffers.

I08—$\overline{I08}$ — Selects which ROM output buffers are enabled.

CEB(XX) — Chip enable, basic machine selects which ROM chips are turned on (+ 12V – ON)

CEA(XX) — Chip enable add-on function.

AEN — Address enable. AEN = RDM + WTM

A05—A09 — Address bits for R/W memory. (+16V & GND)

CSB(XX) — Chip select, basic machine a negative true clock which selects which R/W chips are turned on. (+16V & GND)

CSA(XX) — Chip select, add on.

RWE — Read/Write enable. A +5V signal any time a R/W chip is addressed for a machine memory cycle.

MAW — Max addressable work (+5V signal whenever addressing memory location octal 1377).

NVA — Non-valid address +5v signal for all addresses greater than octal 22000.

$\overline{RWA}$ — Read/Write add on. Whenever addressing R/W memory between octal 20000 & 22000 0v = True.

$\overline{RWS}$ — Read/Write Basic. Whenever addressing R/W memory between octal 1400 & 1777 or octal 16400 & 17777.

OTHER SIGNALS USED BY THE M-REGISTER

Name	Source
T-BUS	PROCESSOR
SCK	"
$\overline{RDM}$	"
$\overline{WTM}$	"
$\overline{TTM}$	"
VLD	CONTROL
MTS	"
CSL	"
$\overline{REF}$	"

SIGNALS GENERATED ON CONTROL BOARD

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). (TTT) + (TOO). (TTS)

MTS — M-Register to $\overline{S}$-BUS. Generated from SC0, $\overline{SC1}$.

A00—A04 — Address bits for R/W memory also used during memory refresh.

$\overline{TSC}$ — T-Register series clock. $\overline{TSC} = \overline{SCK}$. $(\overline{TTS} + \overline{TTT} + \overline{EDT})$

TPC — T-Register parallel clock. (Strobes in data from memory) only during internal memory read cycle.

YBL — Y-enable. A clock which left at + 16V for a read and clocked to GND during a write. (R/W memory only)

PCG — Precharge. The 3rd 16V clock required by the 1103 R/W memory chips.

CLS — Chip select clock. The signal which is anded with M-register data to provide chip selects.

$\overline{REF}$ — Refresh. OV when the memory is in a refresh cycle.

$\overline{CEM}$ — 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. 0v = True.

$\overline{S}$-BUS — Gates I/O register, data, TØØ, MØØ or ones onto $\overline{S}$-BUS and sent to processor. O = True.

$\overline{INH}$ — 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.

$\overline{TMC}$ — 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.

$\overline{EMC}$ — Extended memory cycle. +5v signal used to signal extended memory to begin its cycle. 0v = True.

VLD — A signal generated half way thru memory cycle to allow data to flow out of ROM. Same timing as VOR.

OTHER SIGNALS USED BY THE CONTROL

Name	Source
$\overline{CCT}$	PROCESSOR
TOO	T-REG
IOD	PROCESSOR
ITS	"
SCO	"
SCI	"
$\overline{TTT}$	"
T-BUS	"
SCK	"
$\overline{RDM}$	"
$\overline{WTM}$	"
MCK	"
M00—M04	M-REG
AEN	M-REG
$\overline{EDT}$	EXTENDED MEMORY
D-BUS	I/O CONNECTOR
NVA	M-REG
$\overline{AOF}$	ADD-ON R/W MEMORY (Option 01)
$\overline{EMB}$	EXTENDED MEMORY
MAW	M-REG
$\overline{RWA}$	M-REG

SIGNALS GENERATED ON ROM BOARD

ROD(XX) — Read only Data

OTHER SIGNALS USED BY ROM

Name	Source
CEA(XX)	M-REG
CEB(XX)	M-REG
I00—I07	M-REG
I08 $\overline{I08}$	M-REG
VOR	CONTROL

OUTPUT DISPLAY UNIT

Referring to Figures 15A—D, there is shown the hardware associated with the calculator display. The display comprises a single register 400 of sixteen alphanumeric characters, each character position of which is a seven row by five column matrix of light emitting diodes (LEDs). Register 400 is divided into four quadrants 402 of four character positions each. In addition to the display hardware illustrated, the complete calculator display system comprises bit patterns stored in the calculator ROM, a firmware display routine, and an I/O register, all of which are described in detail elsewhere in this specification. The firmware display routine reads a particular bit pattern, generates a column address 406 to position the bit pattern on the display register 400, and transfers the bit pattern and column address to the I/O register. The display hardware then decodes the column address and turns on each LED as selected by the bit pattern. This process occurs once for each scan of each of the eighty columns of the display. The complete display is scanned approximately sixty times per second.

Input required for the display hardware are: seven bits of character data 404, a seven-bit column address 406, and an enable signal 408. These signals are positive true logic except for the enable signal 408 which must be zero to enable the display.

The character dats 404 is applied through character data inverters 410 to each of four sets of row drivers 412 (eg. row one character data is applied to all four row one drivers, etc.). The five least significant bits of the column address 406 are decoded by the 1-of-20 decoder 414 and turn on one of the column drivers 416. The two most significant bits of column address 406 are applied to the 1-of-4 decoder 418 which turns on one of the quadrant gates 420 when the enable signal 408 is received. The LEDs at the intersections of rows selected by character data 404 and columns selected by column address 406 are then forward biased and emit light.

The 1-of-4 decoder 418 has a fourth input fed by the retriggerable monostable multivibrator 412. In the event the enable signal 408 remains on for approximately five hundred microseconds, the retriggerable monostable multivibrator 422 changes state and turns off the 1-of-4 decoder 418, thereby acting to protect the LED matrix and other circuit components from high DC currents.

KEYBOARD INPUT UNIT, ETC.

The keyboard input unit 12, magnetic card reading and recording unit 14, output printer unit 20, input-output control unit 60, register unit 114, arithmetic logic unit 116, programmable clock 118 and microprocessor 120 are constructed in the same manner as shown and described in the complete specifications of British Patent Application Nos. 1403703, 1403704, 1403705, 1403706, 1403707, 1403708, 1403709, 1403710.

WHAT WE CLAIM IS:—

1. An electronic programmable self-contained desk top calculator including keyboard input means having the capability of entering alphameric information in the form of numerals, algebraic (as defined) expressions and alphabetic messages into the calculator; memory means for storing information entered into the calculator; processing means for processing information entered into the calculator or stored in said memory means to perform selected functions; and output means for providing an output indication of selected functions performed by the calculator; said keyboard input means including a plurality of keys which, in use, in the operation of the calculator enter lines of one or more alphameric statements (as defined) each into the calculator, an execute control key which, in use, in the operation of the calculator, initiates execution of a line of one or more alphameric statements by said processing means, and a store control key which, in use, in the operation of the calculator, initiates storage of a line of one or more alphameric statements in said memory means; said calculator further including buffer storage means coupled to said keyboard input means for storing each line of one or more alphameric statements as it is entered into the calculator from said keyboard input means; said memory means being coupled to said buffer storage means for storing lines of one or more alphameric statements each entered into the calculator; and said processing means being coupled to said keyboard input means, buffer storage means, and memory means, being responsive to actuation of said execute control key, following entry of a line of one or more alphameric statements into said buffer storage means, for executing that line of one or more

OUTPUT DISPLAY UNIT

Referring to Figures 15A—D, there is shown the hardware associated with the calculator display. The display comprises a single register 400 of sixteen alphanumeric characters, each character position of which is a seven row by five column matrix of light emitting diodes (LEDs). Register 400 is divided into four quadrants 402 of four character positions each. In addition to the display hardware illustrated, the complete calculator display system comprises bit patterns stored in the calculator ROM, a firmware display routine, and an I/O register, all of which are described in detail elsewhere in this specification. The firmware display routine reads a particular bit pattern, generates a column address 406 to position the bit pattern on the display register 400, and transfers the bit pattern and column address to the I/O register. The display hardware then decodes the column address and turns on each LED as selected by the bit pattern. This process occurs once for each scan of each of the eighty columns of the display. The complete display is scanned approximately sixty times per second.

Input required for the display hardware are: seven bits of character data 404, a seven-bit column address 406, and an enable signal 408. These signals are positive true logic except for the enable signal 408 which must be zero to enable the display.

The character dats 404 is applied through character data inverters 410 to each of four sets of row drivers 412 (eg. row one character data is applied to all four row one drivers, etc.). The five least significant bits of the column address 406 are decoded by the 1-of-20 decoder 414 and turn on one of the column drivers 416. The two most significant bits of column address 406 are applied to the 1-of-4 decoder 418 which turns on one of the quadrant gates 420 when the enable signal 408 is received. The LEDs at the intersections of rows selected by character data 404 and columns selected by column address 406 are then forward biased and emit light.

The 1-of-4 decoder 418 has a fourth input fed by the retriggerable monostable multivibrator 412. In the event the enable signal 408 remains on for approximately five hundred microseconds, the retriggerable monostable multivibrator 422 changes state and turns off the 1-of-4 decoder 418, thereby acting to protect the LED matrix and other circuit components from high DC currents.

KEYBOARD INPUT UNIT, ETC.

The keyboard input unit 12, magnetic card reading and recording unit 14, output printer unit 20, input-output control unit 60, register unit 114, arithmetic logic unit 116, programmable clock 118 and microprocessor 120 are constructed in the same manner as shown and described in the complete specifications of British Patent Application Nos. 1403703, 1403704, 1403705, 1403706, 1403707, 1403708, 1403709, 1403710.

WHAT WE CLAIM IS:—

1. An electronic programmable self-contained desk top calculator including keyboard input means having the capability of entering alphameric information in the form of numerals, algebraic (as defined) expressions and alphabetic messages into the calculator; memory means for storing information entered into the calculator; processing means for processing information entered into the calculator or stored in said memory means to perform selected functions; and output means for providing an output indication of selected functions performed by the calculator; said keyboard input means including a plurality of keys which, in use, in the operation of the calculator enter lines of one or more alphameric statements (as defined) each into the calculator, an execute control key which, in use, in the operation of the calculator, initiates execution of a line of one or more alphameric statements by said processing means, and a store control key which, in use, in the operation of the calculator, initiates storage of a line of one or more alphameric statements in said memory means; said calculator further including buffer storage means coupled to said keyboard input means for storing each line of one or more alphameric statements as it is entered into the calculator from said keyboard input means; said memory means being coupled to said buffer storage means for storing lines of one or more alphameric statements each entered into the calculator; and said processing means being coupled to said keyboard input means, buffer storage means, and memory means, being responsive to actuation of said execute control key, following entry of a line of one or more alphameric statements into said buffer storage means, for executing that line of one or more

* C L M S *

alphameric statements, and being responsive to said store control key, following entry of a line of one or more alphameric statements into said buffer storage means, for storing that line of one or more alphameric statements in said memory means.

2. An electronic calculator as in claim 1 wherein said keyboard input means includes recall means for recalling a designated line of one or more alphameric statements from said memory means to said buffer storage means; and said processing means is responsive to actuation of said execute control key, following recall of a line of one or more alphameric statements to said buffer storage means, for executing that line of one or more alphameric statements, and is responsive to actuation of said store control key, following recall of a line of one or more alphameric statements into said buffer storage means, for storing that line of one or more alphameric statements into said buffer storage means.

3. An electronic calculator as in claim 1 or 2 wherein said output means is coupled to said buffer storage means and processing means for visually displaying a line of one or more alphameric statements stored in said buffer storage means.

4. An electronic calculator as in any of the preceding claims wherein said processing means includes means for storing routines and subroutines to be performed by the calculator in executing and storing lines of one or more alphameric statements each; and means responsive to actuation of said execute control key for selectively performing one or more of said routines and subroutines to execute the line of one or more alphameric statements then stored in said buffer storage means, and responsive to actuation of said store control key for selectively performing one or more said routines and subroutines to store the line of one or more alphameric statements then stored in said buffer storage means in said memory means.

5. An electronic programmable self-contained desk top calculator including keyboard input means having the capability of entering alphameric information in the form of numerals, algebraic (as defined) expressions and alphabetic messages into the calculator; memory means for storing information entered into the calculator; processing means for processing information entered into the calculator or stored in said memory means to perform selected functions; and output means for providing an output indication of selected functions performed by the calculator; said keyboard input means including a plurality of keys which, in use, in the operation of the calculator, enter lines of one or more alphameric statements each into the calculator, an execute control key which, in use, in the operation of the calculator, initiates execution of a line of one or more alphameric statements by said processing means, and a store control key which, in use, in the operation of the calculator, initiates storage of a line of one or more alphameric statements in said memory means; said calculator further including buffer storage means coupled to said keyboard input means for storing each line of one or more alphameric statements as it is entered into the calculator from said keyboard input means; said memory means being coupled to said buffer storage means for storing lines of one or more alphameric statements each entered into the calculator; said processing means being coupled to said keyboard input means, buffer storage means, and memory means, being responsive to actuation of said execute control key, following entry of a line of one or more alphameric statements into said buffer storage means, for executing that line of one or more alphameric statements, and being responsive to actuation of said store control key, following entry of a line of one or more alphameric statements into said buffer storage means, for storing that line of one or more alphameric statements in said memory means; and said output means comprising alphameric display means coupled to said buffer storage means and processing means for visually displaying a line of one or more alphameric statements stored or being stored in said buffer storage means to enable the user to check for errors therein during and after entry of that line.

6. An electronic calculator as in claim 5 wherein said keyboard input means includes a first control key operable with one or more other keys, in the operation of the calculator, to designate a particular line of one or more alphameric statements stored in said memory means, and a second control key which, in use, in the operation of the calculator nondestructively recalls a designated line of one or more alphameric statements from said memory means to said buffer storage means; and said processing means is responsive to actuation of said second control key, following actuation of said first control key and one or more other keys designating a particular line of one or more alphameric statements stored in said memory means, for nondestructively transferring that line of one or more

alphameric statements from said memory means to said buffer storage means to enable the user to observe a display thereof by said alphameric display means.

7. An electronic calculator as in claim 6 wherein said processing means is responsive to successive actuations of said second control key for transferring successive lines of one or more alphameric statements each from said memory means to said buffer storage means to enable the user to observe a visual display thereof by said alphameric display means.

8. An electronic calculator as in either one of claims 6 and 7 wherein a separate line number is associated with each line of one or more alphameric statements stored in said memory means; and said processing means includes an interpreter routine responsive to actuation of said first control key, followed by actuation of one or more other keys representing a particular line number, for computing the current address in said memory means of the line of one or more alphameric statements associated with that line number, and an uncompiler routine responsive to actuation of said second control key for nondestructively transferring the line of one or more alphameric statements specified by that address from said memory means to said buffer storage means.

9. An electronic calculator as in any one of the preceding claims 5 to 8 wherein said keyboard input means includes a plurality of editing keys which, in use, in the operation of the calculator, edit a line of one or more alphameric statements, entered into said buffer storage means from said keyboard input means or recalled thereto from said memory means, without altering the contents of said memory means; and said processing means is responsive to actuation of said store control key, following editing of a line of one or more alphameric statements stored in said buffer storage means, for storing that edited line in said memory means.

10. An electronic calculator as in claim 9 wherein said processing means is responsive to actuation of said store control key, following editing of a line of one or more alphameric statements recalled to said buffer storage means from said memory means, for storing that edited recalled line back into said memory means in place of the corresponding unedited line previously stored therein.

11. An electronic calculator as in either one of claims 9 and 10 wherein said editing keys include a forward control key and a backward control key which, in use, in the operation of the calculator single step either forward or backward through a line of one or more alphameric statements entered into said buffer storage means from said keyboard input means or recalled from said memory means; and said alphameric display means is operable for visually displaying part or all of the entered or recalled line of one or more alphameric statements up to and including as the last displayed character the last character to which that line has been single stepped.

12. An electronic calculator as in claim 11 wherein said buffer storage means includes a pointer, said pointer being responsive to actuations of either said forward control key or said backward control key for single stepping either forward or backward, respectively, through a line of one or more alphameric statements stored in said buffer storage means and for designating the last character in that line to be displayed by said alphameric display means.

13. An electronic calculator as in either one of claims 11 and 12 wherein said processing means includes routines responsive to successive actuations of said forward and backward control keys for single stepping character-by-character through a line of one or more alphameric statements recalled to said buffer storage means from said memory means and, upon reaching one end of that recalled line, for recalling the next succeeding or preceding line of one or more alphameric statements stored in said memory means to said buffer storage means.

14. An electronic calculator as in any one of the preceding claims 11 to 13 wherein said processing means includes a replacement routine for conditioning the calculator to allow replacement of one or more characters of a line of one or more alphameric statements, entered into said buffer storage means from said keyboard input means or recalled from said memory means, by actuating keys of said keyboard input means that are representative of the desired one or more replacement characters, following actuation of at least one of said forward and backward control keys for single stepping to the desired one or more replacement characters, following actuation of at least one of said forward and backward control keys for single stepping to the desired one or more characters to be replaced.

15. An electronic calculator as in any one of the preceding claims 11 to 14

* C L M S *

wherein said editing keys include a delete control key which, in use, in the operation of the calculator, selectively deletes one or more characters of a line of one or more alphameric statements, entered into said buffer storage means from said keyboard input means or recalled thereto from said memory means, following actuation of at least one of said forward and backward control keys for single stepping to the one or more characters to be deleted.

16. An electronic calculator as in claim 15 wherein said processing means includes a delete routine responsive to actuation of said delete control key, following actuation of at least one of said forward and backward control keys for single stepping to a particular character of a displayed line of one or more alphameric statements, for deleting that character.

17. An electronic calculator as in any one of the preceding claims 6 to 8 wherein said keyboard input means includes a delete control key which, in use, in the operation of the calculator, deletes from said memory means a line of one or more alphameric statements recalled from said memory means to said buffer storage means.

18. An electronic calculator as in any one of the preceding claims 11 to 17 wherein said editing keys include an insert control key which, in use, in the operation of the calculator, inserts one or more characters at selected positions within a line of one or more alphameric statements, entered into said buffer storage means from said keyboard input means or recalled thereto from said memory means, following actuation of at least one of said forward and backward control keys for single stepping to the selected position.

19. An electronic calculator as in claim 18 wherein said processing means includes an insert routine responsive to actuation of said insert control key, following actuation of at least one of said forward and backward control keys for single stepping to a particular position of a displayed line of one or more alphameric statements, for inserting one or more additional characters.

20. An electronic calculator as in any of the preceding claims 6 to 8 wherein said keyboard input means includes an insert control key which, in use, in the operation of the calculator, inserts one or more lines of one or more alphameric statements each between any two designated lines of one or more alphameric statements stored in said memory means.

21. An electronic calculator as in claims 20 wherein said processing means includes routines responsive to actuation of said insert control key, following recall of a line of one or more alphameric statements from said memory means to said buffer storage means, for inserting one or more lines of one or more alphameric statements each into said memory means immediately following that recalled line.

22. An electronic calculator according to claim 1 wherein said processing means is operable, during entry of lines of one or more alphameric statements each into the calculator, for detecting syntax errors in the entered statements as such errors occur; and said output means is responsive to detection of a syntax error by said processing means for providing a visual indication thereof to the user.

23. An electronic calculator as in claim 22 wherein said processing means is operable, upon detection of a syntax error in a line of one or more alphameric statements being entered into the calculator, for inhibiting actuation of any key of said keyboard input means which is not effective in determining, removing, or correcting that error.

24. An electronic calculator as in either one of claims 22 and 23 wherein said processing means and output means are operable for visually displacing an alphameric message indicating to the user the occurrence of each syntax error.

25. An electronic calculator as in any one of the preceding claims 22 to 24 wherein said keyboard input means includes a control key; said output means comprises alphameric display means for visually displaying alphameric information as it is being entered into the calculator; and said processing means is operable, upon detection of a syntax error during entry of a line of one or more alphameric statements into the calculator, for replacing the alphameric information then displayed by an alphameric indication of the syntax error, and is thereafter responsive to actuation of said control key for causing said alphameric display means to again display the previously replaced alphameric information, including the erroneous keyboard entry.

26. An electronic calculator as in any one of the preceding claims 22 to 24 wherein said keyboard input means includes a control key; said ouput means comprises alphameric display means for visually displaying alphameric



information as it is being entered into the calculator; and said processing means is operable, upon detection of a syntax error during entry of a line of one or more alphameric statements into the calculator, for replacing the alphameric information then displayed by an alphameric indication of the syntax error, and is thereafter responsive to actuation of said control key for causing said alphameric display means to again display the previously replaced alphameric information up to but not including the erroneous keyboard entry.

27. An electronic calculator according to claim 1 wherein said keybaord input means includes a programme execution control key which, in use, in the operation of the calculator, initiates execution of one or more lines of alphameric information stored as a programme in said memory means; and said processing means is responsive to an error encountered during execution of a line of alphameric information entered into the calculator from said keyboard input means or stored in said memory means for providing an indication of that error to the user.

28. An electronic calculator as in claim 27 wherein a separate line number is associated with each line of alphameric information stored in said memory means; and said processing means is responsive to an error encountered during execution of a line of alphameric information stored in said memory means for providing an indication of the line number associated with that line.

29. An electronic calculator as in either one of claims 27 and 28 wherein said output means comprises alphameric display means for visually displaying to the user an indication of each error encountered during execution of a line of alphameric information and, in the case of an error encountered during execution of a line of alphameric information stored in said memory means, an indication of the line number associated with that line.

30. An electronic calculator as in any one of the preceding claims 27 to 29 wherein said keyboard input means includes a recall control key; and said processing means is responsive to actuation of said recall control key alone or with other keys following occurrence of an error during execution of a line of alphameric information for causing said output means to display the line, or at least that portion thereof, during execution of which the error occurred.

31. An electronic calculator according to claim 1 wherein each of said alphameric keys is associated with a different representation of the alphameric information entered into the calculator by pressing that alphameric key; said output means comprises alphameric display means, coupled to said buffer storage means, for visually displaying the representations of the alphameric information stored in said buffer storage means; and said processing means is coupled to said alphameric display means and is responsive to actuation of each alphameric key employed in entering a line of alphameric information representing an algebraic expression and/or an alphabetic message into the calculator for causing said alphameric display means to display the representation associated with that alphameric key.

32. An electronic calculator as in claim 31 wherein said alphameric display means is capable of simultaneously visually displaying representations of an amount of alphameric information of shorter length than said buffer storage means is capable of storing; and said processing means is automatically operable, when a line of alphameric information being stored in said buffer storage means exceeds said amount of alphameric information, for causing said alphameric display means to display the representations of said amount of alphameric information last stored in said buffer storage means.

33. An electronic calculator as in claim 32 wherein said keyboard input means includes one or more display control keys which, in use, in the operation of the calculator, control said alphameric display means; and said processing means is responsive to actuation of said one or more display control keys which, in use, in the operation of the calculator, cause said alphameric display means to display the representations of any desired portion of a line of alphameric information stored in said buffer storage means.

34. An electronic calculator as in any one of the preceding claims 31 to 33 wherein said keyboard input means includes a clear control key which, in use, in the operation of the calculator initiates the clearing of said buffer storage means and said alphameric display means; and said processing means is responsive to actuation of said clear control key for clearing said buffer storage means and said alphameric display means without affecting the alphameric information stored in said memory means.

35. An electronic 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.