A MODIFIED BASIC COMPILER IN PL/I FOR THE M6800 MICROPROCESSOR

by

Fred Esenwein

Submitted in Partial Fulfillment of the Requirements
for the Degree of
Master of Science in Engineering
in the
Electrical Engineering
Program

3-25-80
Adviser Date

3-24-80
Dean of the Graduate School Date

YOUNGSTOWN STATE UNIVERSITY

March, 1980

ABSTRACT

A MODIFIED BASIC COMPILER IN PL/I
FOR THE M6800 MICROPROCESSOR

Fred Esenwein
Master of Science in Engineering
Youngstown State University, 1980

This paper presents a compiler which accepts a modified version of BASIC programming language as input, and produces hexadecimal assembly code for the Motorola M6800 microprocessor as output.

Included are discussions of microprocessor design considerations, compiler architecture, and the compiler program itself.

Finally, as a demonstration, the control of a real system is implemented by means of a microprocessor controller for which the operating program is written in BASIC.

ACKNOWLEDGEMENTS

I am obliged to my faculty adviser, Dr. Robert H. Foulkes, for his part in bringing this thesis to completion.

Thanks, also, to General Motors Corporation for financial support during the course of my study.

Most of all, I am grateful to my wife, Mary, whose forbearance throughout the two year span of this project was surpassed only by her expert technical assistance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Purpose and Preliminary Concepts

It is difficult to overstate the impact of recent developments in microprocessor technology upon industrialized society. These compact, relatively inexpensive computer systems are finding applications in such diverse areas as business, education, transportation, and entertainment. The advent of the microcomputer has particularly benefitted the field of industrial control.

Dedicated control functions which would have previously been accomplished using hard-wired logic are now being implemented with microprocessor controllers. This results in a substantial savings in design time, provided that the necessary software can be developed expeditiously. A technique is presented here which simplifies the development of software for the Motorola M6800 microprocessor.

Microprocessor machine code is a collection of hexadecimal words that, in program form, is nearly useless to all but the most experienced programmer. A mnemonic form is available, but if it must be assembled by hand using pencil and paper, the process is slow and tedious. It is desirable to write the source program in a higher level

language like BASIC or FORTRAN, and then invoke a machine translation of that source program into hexadecimal codes required by the microprocessor. Such packages are available from major manufacturers of microprocessor products as software support for their development systems. Unfortunately, these development systems are expensive enough to be unaffordable for many small businesses and all but a few individuals. This paper presents a compiler in PL/I which translates BASIC source code into M6800 machine language. Anyone with access to a time-sharing host computer is thereby able to develop microprocessor based control system software in BASIC language.

Approach

Industrial control requirements vary widely. In many instances intelligent control is needed, but the computing power of either a mainframe computer or a minicomputer would constitute an overkill situation. It is to these applications that the microcomputer is ideally suited. As a simple dedicated controller, the microcomputer is not required to perform complex mathematical operations. It is often not interfaced with a line printer. These and other simplifying factors lead to modification of the BASIC programming language for purposes of dedicated control. A later section will deal with the development of this modified BASIC language.

Although any of several microprocessors could be programmed in BASIC by means of a compiler similar to the one presented here, the Motorola M6800 was chosen in this particular case. The M6800 is an industry standard, and is supported by a wide variety of peripheral devices. In addition, a microcomputer built around the M6800 CPU was available to the author at the time this project began. Thorough testing of algorithms throughout the development of the compiler was therefore possible.

The compiler itself is divided into three separate parts: lexical analysis, syntax analysis, and code generation. The lexical analysis phase breaks each line of source code down into its component parts and checks for elementary errors. Syntax analysis discerns from the order of words and phrases in the source code what action must be performed by the program. These actions are translated into microprocessor machine code by the code generator.

The chapters which follow discuss the details involved in each segment of the compiler and how they accomplish the translation of BASIC into M6800 code. Chapter I has explained the purpose and basic approach for this paper. Chapter II deals with the peculiarities of the M6800 microprocessor and BASIC programming language. In Chapter III, the compiler proper is discussed, while Chapter IV presents results. A user's manual, PL/I listings for the compiler, sample programs, a demonstration program, and a glossary of compiler terms constitute the five appendices.

CHAPTER II

DESIGN CONSIDERATIONS

M6800 Hex Code as Compiler Output

Prerequisite to the design of a compiler that produces proper hex code for the M6800 microprocessor is an understanding of the features, limitations, and general operation of the machine itself. A cursory examination of M6800 fundamentals is therefore in order.

A program for the M6800 consists of a series of instructions, each of which is followed by one or more operands. There are over one hundred hexadecimal instructions (opcodes) for the 6800 MPU [1]. It is the task of the compiler to express a source program, which is written in BASIC, solely in terms of this instruction set.

Many instructions cause the MPU to either store data in memory or to retrieve data from memory. When this is the case, the programmer often has the choice of addressing the desired memory location in one of several ways. The four possible choices are the immediate, direct, indexed and extended addressing modes. Efficient use of available memory and speed of execution can be maximized by the proper choice of addressing mode.

When an instruction is executed in the immediate addressing mode, the byte immediately following the opcode

is used as data. This is useful when the data does not change during the execution of the program.

Direct addressing interprets the byte following an opcode as a pointer to the memory location from which the data for the operation is to be retrieved. Since the 6800 MPU uses a 16 bit address bus, a specific memory location can not be completely defined by one eight bit byte. Direct addressing assumes the higher order byte to be 00_{16}. Direct addressing is sometimes called zero page addressing.

The use of extended addressing allows any memory location to be accessed via the 16 bit address bus. An opcode is followed by two bytes in the program, the first of which is the higher order byte of the address where the data resides, and the second of which is the lower order byte. This mode of addressing requires more execution time, but it is more flexible than immediate or zero page modes.

Indexed addressing will not be discussed in this paper.

The need for an understanding of various addressing modes will become apparent later when memory assignment is discussed.

Although the 6800 MPU is capable of addressing over 64,000 memory locations, very few systems actually incorporate that much memory. Indeed, memory size is a limiting factor in almost all microprocessor system designs. The machine used by the author is marketed by Heath Company. It has only 512 bytes of random access memory (RAM). Of this,

addresses 00C5 through 00FF are reserved for use by the operating system [2]. This lack of memory places severe restrictions on the configuration of a BASIC compiler to be used with this machine. The size of the BASIC source program acceptable for compilation is likewise restricted.

Input/output (I/O) devices are assigned addresses by the system designer. Transfers of data between I/O devices and the MPU are handled exactly like memory transfers. Generally speaking, I/O devices are assigned addresses above the highest address assigned to a RAM element.

This completes the overview of microprocessor fundamentals relevant to the development of the BASIC compiler. A detailed treatment of this subject is outside the scope of this paper. Readers who are not familiar with the operation of the Motorola M6800 should seek a good tutorial such as the Heathkit individual learning program [1].

As stated previously, the output of the BASIC compiler is a program in hexadecimal codes that are part of the M6800 instruction set or data. This can be enhanced for ease of human interpretation by the addition of mnemonic instructions and comments in English. These concepts are incorporated into the finished BASIC compiler.

Choice of BASIC as Source Language

Although no single high level computer language is best for all applications, BASIC seems to be a good choice in this instance. It is easily learned and universally used.

In addition, with minor modifications, it is ideally suited for industrial control functions.

Many versions of BASIC are currently in use. Some are so expanded with respect to the original Dartmouth BASIC that they approach FORTRAN or PL/I in complexity. Table 1 shows commonly used elements of BASIC language as derived from Digital Equipment Corporation's BASIC-PLUS [3].

TABLE 1

FUNDAMENTAL ELEMENTS OF BASIC

Operators	Functions	Statements
-	ABS	LET
+	ATN	PRINT
/	COS	GOTO
*	DEF	INPUT
↑	EXP	STOP
=	INT	END
(	LOG	IF
)	RND	FOR
>	SGN	NEXT
<	SIN	GOSUB
,	SQR	RETURN
>=	TAN	DIM
<=		REM
<>		DATA
		READ
		RESTORE
		ON
		matrix operations
		alphanumeric operations
		file handling operations

There are many items in the above listing that serve no useful purpose as far as dedicated industrial control is concerned. On the other hand, there are some instructions

that would be very nice to have which do not appear above. The task at hand becomes one of deleting those statements, functions, or operators which are either meaningless or worthless to the intended application, and of adding special statements, functions, or operators to facilitate working with the microprocessor.

First of all, the PRINT statement is meaningless without a printer. While data manipulation on a mainframe computer almost invariably results in a report from the system printer, printers are often left out of microprocessor systems altogether. When they are incorporated, they may be treated just like any other I/O device. The PRINT statement is therefore of little use and will be deleted.

Similarly, the alphanumeric and quoted text features serve a limited usefulness and are also deleted.

The INPUT statement is a system command. It is used in conjunction with a terminal for interactive processing. Any time the INPUT statement is encountered in the execution of a BASIC program, the execution halts and the user is prompted to enter a value at the terminal. Interactive processing is not a concern here, so the INPUT statement is deleted.

File handling statements are nice for organizing data in applications like payroll management or license plate registrations. They are not necessary in this case.

Matrix operations have been dropped for two reasons. First of all, matrices require tremendous amounts of storage.

Secondly, matrix operations are of very little use in a real time control system. In most cases the real time event would be long past by the time the matrix operations were completed.

All of the BASIC functions are deleted with the exception of SQR. SQR is retained as an academic exercise in the development of functional algorithms.

All of the operators are retained.

The statement list is now down to a much more manageable size. At this point, however, there exists a total lack of I/O statements for peripheral devices. This, of course, is a direct result of the elimination of the PRINT and INPUT commands. They must now be replaced by more general I/O statements.

The problem with the PRINT and INPUT statements is that they address only two devices, whose addresses are known to the mainframe computer by design. The microprocessor must be able to accept data from or transmit data to a multitude of devices at arbitrary addresses. This is accomplished by defining two new BASIC statements, RDIN and WRTOUT. These statements, in conjunction with the addresses of the appropriate I/O devices, allow the exchange of data between the 6800 MPU and its peripherals. Use and syntax of these new statements are demonstrated in Appendix A.

One more change is necessary. The character "@" must be substituted for the up-arrow operator. No up-arrow exists on the IBM terminal through which this program was

developed.

This concludes the modification of the BASIC language. Table 2 summarizes the results.

TABLE 2

ELEMENTS OF MODIFIED BASIC

Operators	Functions	Statements
-	SQR	LET
+		GOTO
/		STOP
*		END
@		IF
=		FOR
(		NEXT
)		GOSUB
>		RETURN
<		REM
,		DATA
GEQ (>=)		READ
LEQ (<=)		RESTORE
NEQ (<>)		RDIN
		WRTOUT

No attempt to instruct the reader in the use of this modified version of BASIC is made here. A complete user's manual appears in Appendix A.

CHAPTER III

SYNTHESIS OF A COMPILER IN PL/I FOR THE GENERATION OF M6800 HEX CODE FROM BASIC SOURCE PROGRAMS

Compiler Architecture and Operation

A compiler translates a higher level programming language into machine language acceptable to the computer being programmed. This is accomplished in three major steps: (1) lexical analysis, (2) syntax analysis and interpretation, and (3) code generation.

The purpose of lexical analysis is to break the source program down into a series of basic elements (tokens), and to discover elementary errors. Each line of source text is scanned sequentially. Identifiers (variable names), literals (constants), and terminal symbols (operators and keywords) are individually broken out of the source line, checked for validity, and entered into the parse table. The source program is then converted into a series of "uniform symbols". Each entry in the uniform symbol table is three characters long, and is accompanied by a pointer showing where in the tables the actual token may be found. For instance, a uniform symbol table entry like "TRM 4" refers to the fourth token in the terminal table.

The use of uniform symbols makes comparison of long character strings unnecessary. It also makes later phases of the compiler less complicated.

The outputs from the lexical analysis phase are identifier (IDN), literal (LIT), and uniform symbol tables. Lexical analysis also provides input to the error table if mistakes in the source program are detected. The terminal (TRM) and keyword (KEY) tables are permanent tables and are used by lexical analysis as input. All of the above tables, with the exception of the error table, are forwarded as inputs to the syntax analysis phase.

Once through the lexical analysis phase, the compiler has ascertained that the source program is made up of a series of allowable tokens. However, if the program is to have any meaning, the tokens must be arranged in the proper order according to the rules of syntax for the source language.

During syntax analysis the uniform symbol table is scanned sequentially while the computer looks for a meaningful phrase. When a meaningful phrase is recognized, the compiler breaks it down into elementary operations and enters the information into a matrix. Each entry of the matrix consists of a simple operator and one or two operands. In this way complex expressions are reduced to several simpler operations.

As before, if any syntax errors are discovered during the course of the syntax analysis, they are recorded

in the error table.

The matrix is an intermediate form of the program that is derived solely from the source program and is in no way related to the machine for which the compiler is used. In other words, up to this point, the compiler is "machine independent". In the next phase, which is code generation, the output must be tailored for a specific piece of hardware.

The code generator translates each line of the matrix into machine code. The compiler uses a code generation routine dictated by the matrix operator along with the matrix line operands to output a machine language equivalent of the matrix line. Each matrix line may actually result in several lines of machine code.

Compiler theory is the subject of many books. The material in the above discussion was drawn largely from a work by John J. Donovan [4]. The chapter on compilers should be of interest to the reader desiring a deeper understanding of the ideas involved.

The modified BASIC to M6800 hex code compiler, which is the topic of this paper, was developed according to the generalized model of the compiler just presented. The higher level language input is the modified BASIC of Appendix A, and the machine language output is Motorola M6800 hexadecimal code.

The compiler itself is a computer program, written in PL/I which accomplishes the tasks of lexical analysis,

syntax analysis, and code generation on an IBM 370 computer. Because of its structure and ability to perform string operations with relative ease, PL/I was chosen over the other higher level languages available for the development of this compiler.

As previously stated, there are certain areas within the microprocessor's memory of which the compiler may not avail itself when making memory assignments. Figure 1 shows the memory map for the author's system.

SYSTEM ASSIGNMENTS	HEXADECIMAL ADDRESSES	USER ASSIGNMENTS
MONITOR ROM (OPERATING SYSTEM)	FFFF FC00	NOT AVAILABLE
NOT ASSIGNED	FBFF C200	I/O DEVICES
RESERVED FOR KEYBOARD, DISPLAYS, AND MONITOR	C1FF C000	NOT AVAILABLE
NOT ASSIGNED	BFFF 0200	I/O DEVICES
USER RAM	01FF 0100	PROGRAM
RESERVED FOR MONITOR	00FF 00C5	NOT AVAILABLE
USER RAM	00C4 0000	DATA, CONSTANTS, VARIABLES, TEMPORARY STORAGE

Fig. 1. Memory Map.

Addresses 0000 through 01FF are occupied by RAM devices. The system, however, has reserved a section of memory for the microprocessor's operating system in the middle of user RAM. The compiler has therefore been designed to begin address assignments for the actual program at address 0100. If the program were started at a lower address, provision would have to be made for execution to branch around the reserved locations. The addresses 0000 through 00C4 are used by the compiler for the storage of data, constants, variables, and temporary values. These items are frequently transferred to and from the MPU during program execution, so placing them in zero page allows the advantage of using the direct addressing mode.

The remaining memory addresses that are not reserved for system use may be assigned to I/O devices.

The necessary theory and physical constraints have now been brought into focus. The next section deals with the PL/I program that produces M6800 code from modified BASIC input.

Discussion of PL/I Routines

A flow chart of the compiler appears in figure 2. The reader may also wish to refer to the PL/I listings of compiler routines which appear in Appendix B. Each PL/I routine is discussed individually below. The routines are presented in three groups corresponding to the lexical, syntax, and code generation phases of the compiler.

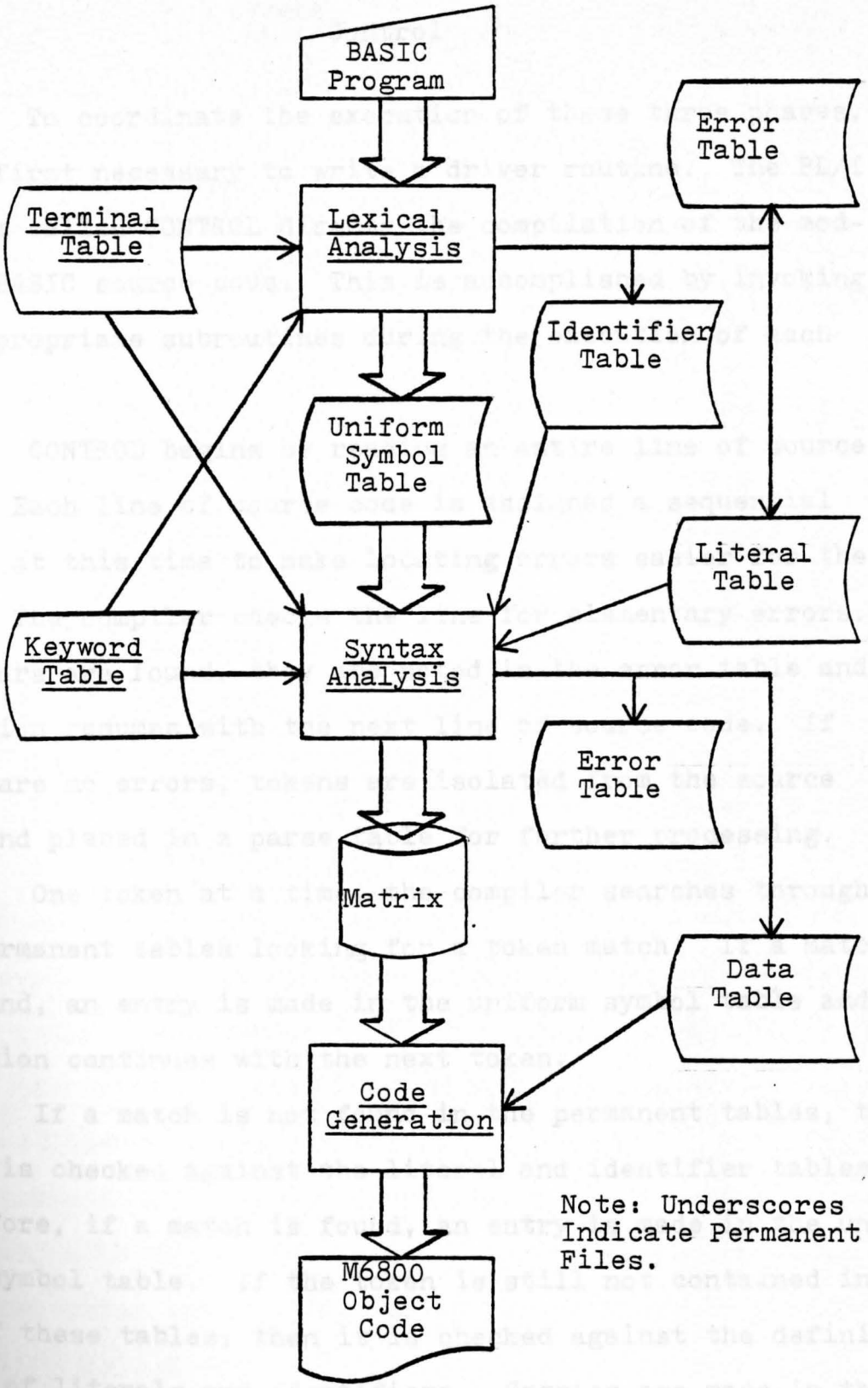

Fig. 2. Modified BASIC to M6800 Hex Code Compiler Flow Chart.

Control

To coordinate the execution of these three phases, it is first necessary to write a driver routine. The PL/I program called CONTROL directs the compilation of the modified BASIC source code. This is accomplished by invoking the appropriate subroutines during the execution of each phase.

CONTROL begins by reading an entire line of source code. Each line of source code is assigned a sequential number at this time to make locating errors easier for the user. The compiler checks the line for elementary errors. If errors are found, they are noted in the error table and execution resumes with the next line of source code. If there are no errors, tokens are isolated from the source line and placed in a parse table for further processing.

One token at a time, the compiler searches through the permanent tables looking for a token match. If a match is found, an entry is made in the uniform symbol table and execution continues with the next token.

If a match is not found in the permanent tables, the token is checked against the literal and identifier tables. As before, if a match is found, an entry is made in the uniform symbol table. If the token is still not contained in any of these tables, then it is checked against the definitions of literals and identifiers. Entries are made in the uniform symbol table and either the literal or identifier table if the token is found to be valid. Otherwise, the

only conclusion is that the token is an illegal character string. This information is sent to the error table.

Execution continues in like manner, one token after another, until all the tokens in the source program have been examined. The result is a complete uniform symbol table, and filled literal and identifier tables with no duplicate entries.

Once the uniform symbol table has been established, the compiler proceeds with syntax analysis, working with one line of source code at a time, as taken from the uniform symbol table.

The first thing which must appear at the beginning of a new line is a valid line number. An invalid line number in the first position causes the generation of an error message. Execution then skips around the remainder of the tokens in that line and begins again at the start of the next source line in the uniform symbol table. In fact, the discovery of an error anywhere within a program line during syntax analysis will result in this same action.

According to the rules of syntax for BASIC, a keyword must always appear in the second position. For every BASIC keyword, there is an associated action routine in the compiler which processes the remainder of the source line. When the syntax analyzer reads the second token in a source line from the uniform symbol table, it checks this token against valid keywords until a match is found. If no match is found, an error is indicated.

A valid keyword in the second position causes execution to transfer to an action routine where the remainder of the line is checked for syntax, interpreted, and finally converted into matrix entries for use by the code generation phase.

The last statement in any BASIC program must be END. The compiler checks for this during syntax analysis and issues an error message if the condition is not met.

Although the entire source program is scanned for lexical and syntax errors, an error of any kind causes compilation to halt at the end of syntax analysis. Code generation is thus suppressed and the programmer is obliged to correct his source program before attempting to compile again.

If the execution continues through the lexical and syntax phases without incident, the final task of code generation is undertaken.

The purpose of the code generation phase is to convert the source program, now in matrix form, into microprocessor hex code and to make memory assignments within user RAM.

The compiler begins at address 0000 and assigns data, literals, identifiers, and temporary storage values to zero page locations. The compiler then begins reading matrix lines, one at a time, and generating equivalent microprocessor code. This microprocessor code is entered in consecutive memory locations, beginning at address 0100.

Code generation works in much the same way as syntax analysis. The compiler reads a matrix line. Each matrix line consists of an operator and one or two operands. Corresponding to every matrix operator is a code generation subroutine. When a matrix operator is recognized, execution is transferred to the associated code generation subroutine where microprocessor code is generated according to the operands in the matrix line.

This completes compilation of the source program. CONTROL sends the source listing, the error table, the terminal table, the literal table, the identifier table, the uniform symbol table, and the data table, along with the finished microprocessor hex code version of the source program to the system printer. The matrix is available in a file called MATRIX DATA.

The discussions which follow deal with the various subroutines used by CONTROL during the course of compilation.

Lexical Phase

Parsing the Source Code

Each line of BASIC source code is resolved into its component parts (tokens), or parsed, by a PL/I routine called PARSE2. PARSE2 scans the source line looking for blanks, terminal symbols, or operators which delineate tokens. The characters between delimiters are assumed to be tokens and are entered into the parse table in the order in which they

are encountered. PARSE2 places a "$" symbol in the parse table at the end of each source line to serve as a line's end flag for the syntax phase.

Discovery of Elementary Errors

ERRCHK examines an entire source line and looks for three error conditions: (1) no numeric in column one, (2) a line number of more than five digits, and (3) characters beyond column 72.

The REMARK Statement

A PL/I routine called REMARK is called by CONTROL whenever a REMARK statement is encountered in the source code. The appearance of "REM" in the parse table when the compiler is building the uniform symbol table causes the compiler to skip to the next program line. No entry is made in the uniform symbol table for any token appearing in the REMARK statement. Anything contained in a REMARK statement is therefore ignored.

Recognition Routines

All tokens must be classified as terminal symbols, keywords, identifiers, or literals. The definitions of these different types of tokens are contained in the recognition routines.

TRMREC, IDNREC, LITREC, KEYREC, and HEXREC are called by CONTROL to classify a token found in the parse

table as a terminal symbol, identifier, literal, keyword, or hexadecimal literal, respectively. Once recognized as a specific type of token, an entry is made for the token in the uniform symbol table.

Syntax Phase

Checking for a Valid Line Number

The routine called TGTCHK checks to see if the first token in a new source line from the uniform symbol table is a valid line (target, for branching) number. To qualify, it must be a literal, and it must be all numeric.

Action Routines

The remainder of the syntax routines are action routines which are called by CONTROL when keywords are encountered in the uniform symbol table. Every keyword in BASIC language indicates a specific set of operations to be performed on the tokens which follow it. The action routine for a given keyword discerns the meaning of the phrase following that keyword according to the rules of syntax for BASIC and outputs a series of matrix lines which convey that meaning to the code generation phase.

All keywords have associated action routines. Included are LETAR, RDINAR, DATAAR, RETRNAR, STOPAR, WTOUTAR, GOTOAR, RESTRAR, READAR, ENDAR, FORAR, NEXTAR, and GOSUBAR. These PL/I routines generate matrix entries for LET, RDIN, DATA, RETURN, STOP, WRTOUT, GOTO, RESTORE, READ, END, FOR,

NEXT, and GOSUB modified BASIC statements, respectively.

The action routines labeled IFAR and CDX are used together to process the IF statement. The IF statement consists of two arithmetic expressions on either side of a relational operator. CDX is used to process the condition on the left hand side of the relational operator and the assignment CXL = CDX is made. Similarly, CDX is used to process the condition on the right hand side of the relational operator and the assignment CXR = CDX is made. Finally, the IF action routine generates matrix entries that compare CXL to CXR prior to making a branch decision.

As before, an error anywhere within the program line results in output to the error table and sends execution to the beginning of the next source line.

Code Generation Phase

Code Generation Routines

The code generation subroutines are used by CONTROL to translate the program as it appears in the matrix into M6800 microprocessor hex code. Each matrix line begins with an operator which, when encountered, sends execution to the code generation subroutine associated with that particular operator. This usually results in several lines of microprocessor code for each matrix line.

The code generation subroutines are easily identified. Each is given a name which begins with the matrix

operator it services, and ends in "CG" or "G". For example, RTRNCG is the code generation subroutine for the RETURN operator.

Table 3 summarizes the PL/I code generation subroutines and the matrix operators with which they are associated.

TABLE 3

CODE GENERATION SUBROUTINES

Code Generation Subroutine	Matrix Operator
READCG	READ
STOPCG	STOP
ENDCG	END
MTPLYCG	* (multiplication)
RESTRCG	RESTORE
EQUALCG	= (equality)
PLUSCG	+ (addition)
MINUSCG	- (subtraction)
DIVIDCG	/ (division)
EXPCG	@ (exponentiation)
SQRCG	SQR (square root)
RDINCG	RDIN
WTOUTCG	WRTOUT
GOSUBCG	GOSUB
GOTOCG	GOTO
RTRNCG	RETURN
BRANCHG	assigns destinations to branch statements
NEQCG	not equal
CONEQCG	conditionally equal
LTCG	less than
GTCG	greater than
GEQCG	greater than or equal
LEQCG	less than or equal

Utility Routines

The remaining PL/I routines are used by the code generation subroutines to accomplish routine, repetitive tasks.

HXDCCON converts a hexadecimal number to a decimal number. DCHXCO and DCHXCO2, on the other hand, convert decimal numbers to hexadecimal equivalents.

HEADING and RITEOUT are used to format the microprocessor code output that is sent to the system printer.

CHAPTER IV

RESULTS

Compiler Output

The report which the user receives from the IBM 370's printer at the completion of compilation consists of two parts: (1) the M6800 microprocessor code version of the source program, which is of primary interest, and (2) supplemental information to aid in the location of a problem, should one arise during compilation. The reader may wish to refer to one of the sample programs in Appendix C for the discussion that follows.

The microprocessor code version of the program is printed out at fifty lines to a page, with each page headed so as to make interpretation of the results easier for the user. In the first column are hexadecimal addresses for memory locations within user RAM. The second column lists the hexadecimal values that are to be programmed into the microprocessor. The first two columns alone are sufficient information for the user to enter and run his program on the M6800 microcomputer.

The last two columns are informatory in nature. Column number three is headed "MNEMONICS/DECIMAL CONTENTS". It contains mnemonics associated with various opcodes, which should be familiar to all experienced microcomputer users.

When helpful for clarity, certain decimal equivalents of hexadecimal constants appear in this column as well.

The last column describes the program in English phrases. This information, along with that in the previous column greatly enhances the program's readability.

When branch statements are encountered during compilation, it often occurs that the address within the microprocessor's memory to which execution is to be sent is at the time unknown. For this reason, all branch statements are saved until the compilation of the source program is complete. At that time, all memory assignments have been made, and the branch statements are listed in one block at the end of the printed output. Meanwhile, on the first pass, the memory locations to be filled with branch statements are simply filled with "XX".

Any time the hexadecimal contents are listed as "XX", it should be understood that the actual contents of that memory location do not matter. This convention is carried over into the locations reserved for computed values such as variable names. Since these locations will be written into later, the actual contents before execution of the microcomputer program can be anything.

Notice that memory addresses are assigned sequentially, beginning with 0000, for data, constants, variables, and temporary storage locations. Memory assignment then skips to address 0100 and continues to the end of the program.

At the end of the program listing, the user is reminded to start execution of his program from address 0100. The total amount of microprocessor memory required for the program is also given, in bytes. This information is very useful to the designer, because he can write a program to perform his control function, compile it, and know exactly how much RAM he must include in his system before actually purchasing any hardware.

Following the M6800 code program is a listing of the modified BASIC source program exactly as it was written by the user. The program lines are numbered consecutively on the right hand side. The error table on the next page of the printed output lists errors, if any, along with source line numbers corresponding to the ones found to the right of the BASIC code.

The last pages of the printed output contain the major tables used or created during compilation. They could easily be deleted from the report, but are included here for completeness. They might also be helpful in locating unusual problems encountered during compilation.

Application to HO Railroad Control

The final test of the results claimed in this paper is to build a working, physical system and control it with a microcomputer, using a control program written in BASIC and translated to microcomputer codes by means of the modified BASIC to M6800 code compiler presented herein.

The system chosen for purposes of demonstration is a model railroad. Parts for this set up are inexpensive and readily available. The track layout is shown in figure 3.

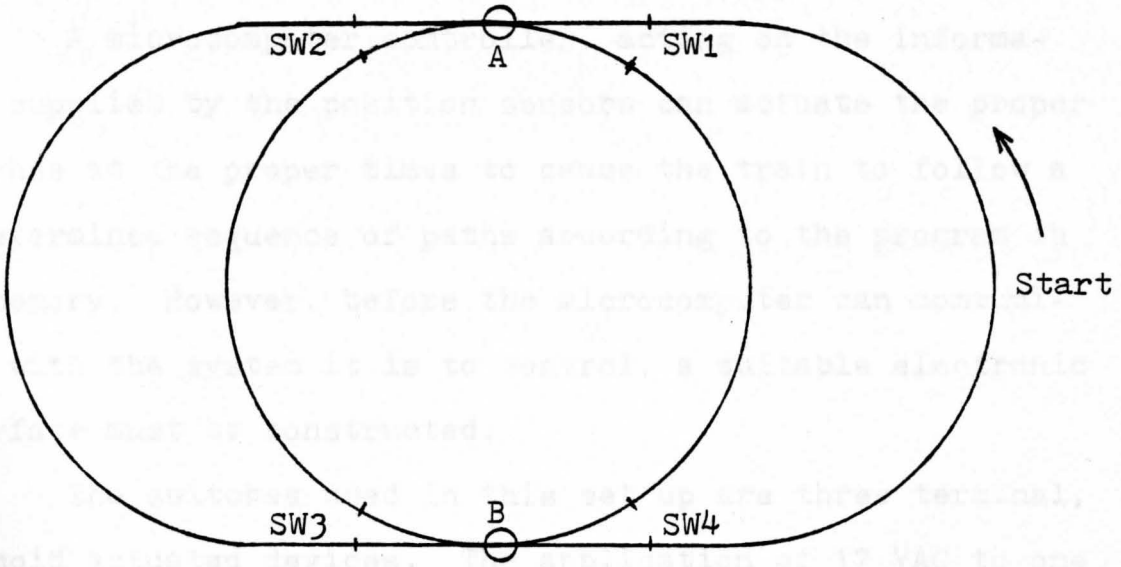

Fig. 3. Configuration of Train Track.

As can be seen in figure 3, a train starting from the location indicated may take any one of four circular paths defined by: (1) the large outside ellipse, (2) the inside circle, (3) the right hand ellipse consisting of the right side of the outside ellipse and the left side of the inside circle, or (4) the left hand ellipse consisting of the left side of the outside ellipse and the right side of the inside circle.

Position sensors are located at points A and B, which are the only two points common to all four possible paths. A cadmium sulfide photoresistor is positioned beneath the track bed at each location, with miniature incandescent lights above. These position sensors act as input

devices.

The output devices are, of course, the four remote control switches labeled SW1, SW2, SW3, and SW4.

A microcomputer controller, acting on the information supplied by the position sensors can actuate the proper switches at the proper times to cause the train to follow a predetermined sequence of paths according to the program in its memory. However, before the microcomputer can communicate with the system it is to control, a suitable electronic interface must be constructed.

The switches used in this set up are three terminal, solenoid actuated devices. The application of 17 VAC to one pair of terminals causes the switch to set for straight through. Voltage applied to the other pair of terminals (one terminal is common in both cases) causes the switch to set for turn out. Three binary combinations are therefore sufficient for driving each switch. We choose them to be 00 for no action, 01 for straight through, and 10 for turn out.

Motorola manufactures an integrated circuit device called the Peripheral Interface Adapter (PIA). It is intended to simplify the chore of interfacing the M6800 microprocessor to physical systems. Coincidentally, the PIA is configured with two I/O ports of eight lines each. Either port can be initialized by the computer as an input port or an output port. This device lends itself perfectly to the application at hand. If we consider all four railroad switches to be one output device, then one eight bit PIA

port can be designated as an output port to drive the four switches, which require two bits each.

The remaining PIA I/O port is defined to be an input port. Two of its eight lines are used to convey position information to the microcomputer.

The schematic diagram of figure 4 shows the interface circuitry used to interconnect the microcomputer and the train layout. The M6820 PIA is labeled IC1. The only other integrated circuits are two common TTL gates.

In the author's system, all of the integrated circuits are located near the microprocessor itself. A ribbon cable connects the PIA to a set of relays and their transistor drivers, which are located on the track bed along with the position sensors.

The hexadecimal address 4006 is assigned to the four railroad switches, and I/O lines PB0 through PB7 are accordingly designated as output lines. The two position sensors are given the address 4004 and are connected to the PIA lines PA0 and PA1, which are defined as input lines. Lines D0 through D7 are connected to the microcomputer data bus.

When the microcomputer executes an instruction to store data in memory location 4006, the information on the data bus is transferred through the PIA to the eight lines which control the railroad switches. The lines having logical ones on them saturate the switching transistors to which they connect, energize the associated relay coils, and apply 17 VAC to the corresponding switch terminals.

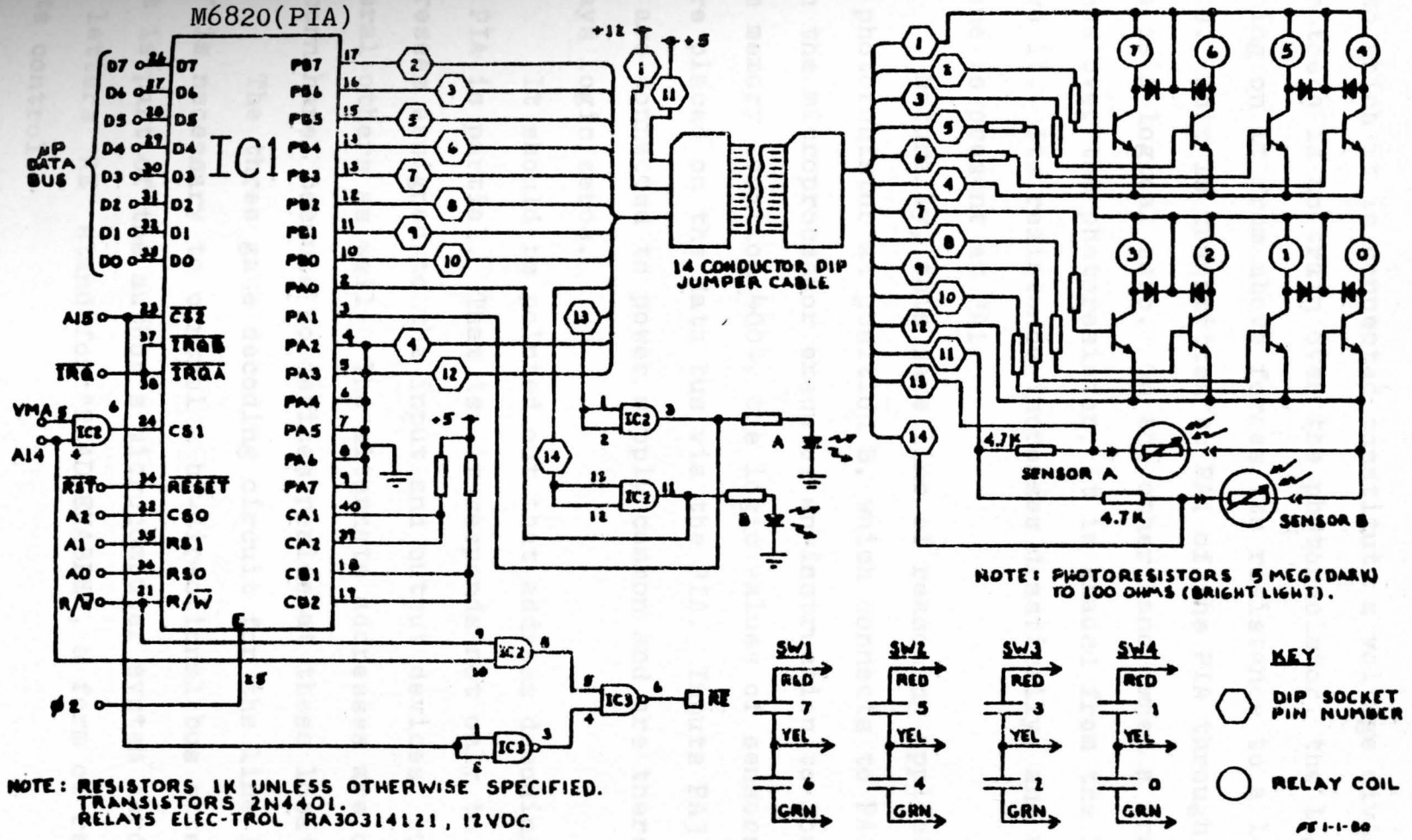

Fig. 4. Interface Circuit Schematic.

The photoresistor at position A and the 4.7K resistor to which it is connected constitute a voltage divider. When there is no train over the photoresistor, the light shining on it from above forces its resistance to a low value. This is transmitted to PA1 of the PIA through the gate as a logical zero. On the other hand, when a train passes over the photoresistor, it is shaded from the light above it. Its resistance increases drastically, and a logical one is present at PA1.

Naturally, this same line of reasoning applies to the photoresistor at position B, which connects to PA0. When the microprocessor executes an instruction to load data from memory location 4004, the logic values of sensors A and B are placed on the data bus via the PIA. Inputs PA3 through PA7 are connected to power supply common and are therefore always logic zeros.

It should be pointed out that address decoding for the PIA is partial. That is, it responds not only to the addresses assigned to the input and output devices, but to several others as well. The alternate addresses are of no concern here, because no devices reside at these locations.

The three gate decoding circuit for the line labeled "$\overline{\text{RE}}$" is necessary to control a bi-directional bus extender that is part of the author's microcomputer system hardware. The letters "RE" stand for "READ ENABLE", a form of read/write control.

Now that the hardware interface has been taken care of, the control program must be approached. Let us start by developing a sequence of control events related to the system's I/O capabilities.

Let us say that we wish the train to take the following path: (1) once around the outside loop, (2) once around the inside circle, (3) once around the right side loop, (4) once around the left side loop, and (5) repeat the above steps until commanded to stop.

The train starts from the position indicated in figure 3. Table 4 outlines the sequence of events that must occur in order to accomplish the above objective.

TABLE 4

SEQUENCE OF EVENTS FOR TRAIN CONTROL

Train Position	SW1	SW2	SW3	SW4	PB Lines of PIA				Decimal Equivalent
					76	54	32	10	
A	X	X	X	X	00	00	00	00	0
B	X	S	X	X	00	10	00	00	32
A	X	X	S	S	00	00	10	10	10
B	S	X	X	X	10	00	00	00	128
A	X	X	X	T	00	00	00	01	1
B	T	X	X	X	01	00	00	00	64
A	X	X	X	S	00	00	00	10	2
B	S	T	X	X	10	01	00	00	144
A	X	X	T	X	00	00	01	00	4
B	X	X	X	X	00	00	00	00	0
A	X	X	X	T	00	00	00	01	1
B	T	X	X	X	01	00	00	00	64

Recall that the switch commands are 00 for no action (X), 01 for straight through (T), and 10 for turn out (S).

The command for each switch is given along with the binary values on each of the PIA output lines for each sequential position of the train. The decimal equivalents of the eight bit binary words are shown in the last column on the right.

The switches must be initialized so that they are all set straight through. Notice that at the end of the sequence they are again all straight through so that the sequence can be repeated indefinitely.

The BASIC program that achieves this control function is presented in Appendix D, along with the resultant M6800 hex code version used to program the microcomputer.

Conclusion

The compiler program presented in this paper provides any user who has access to a mainframe computer hosting PL/I the capability to produce useable M6800 assembly language programs from BASIC source code. The total cost of the demonstration control system, including the microcomputer, the interface components, and the train layout is less than two hundred dollars.

Admittedly, the control function performed by the train controller is not very complex. However, many industrial processes are equally simple. Automation of such processes might be very desirable if they could be done for a reasonable price. Reduction of engineering time through simplified software generation is the best way to reduce the cost of microprocessor control systems.

This paper has developed a compiler for simplifying microprocessor software generation in several logical steps. Chapter I describes the advantages of using microprocessor controllers instead of hard-wired logic and outlines the basic design philosophy behind the compiler itself.

Chapter II provides background information on the M6800 microprocessor, and establishes a special version of BASIC programming language, which is used as source code for the compiler program.

Chapter III is a treatment of compiler architecture in general, and the Modified BASIC to M6800 Hex Code Compiler in particular. All of the PL/I routines constituting the compiler are discussed in this chapter.

Finally, Chapter IV describes the compiler output and how it should be interpreted and used. As proof of the functionality of the compiler program, a model railroad layout is controlled by a microprocessor for which the control program is written in modified BASIC and translated by the compiler into microprocessor code.

Suggestions for expansion of this work include improved program storage capability and interrupt processing. The microprocessor hex code program resulting from compilation is presently stored permanently only on the printed output from the mainframe computer. Program storage on magnetic tape or disk with an appropriate interface to the microcomputer would greatly simplify the task of programming the controller. Also, the compiler presented here does not

allow for the processing of interrupts. This is one of the microprocessor's more powerful features, and might be used to great advantage.

With the addition of sufficient memory to the microcomputer system, functions and special operations could be accomplished using the techniques developed here.

Whether or not the above improvements are undertaken in the future, the compiler as it stands is a useful and functional development tool for anyone operating on a budget.

APPENDIX A

Modified BASIC User's Manual

LANGUAGE SYNTAX

Statements

Keyword: DATA
Type: Non-executable
Format: line number DATA (value list)
Description: Supplies data to READ statement. DATA is never used without READ.

Keyword: END
Type: Executable
Format: line number END
Description: Terminates execution. The end statement is the last statement in a BASIC program.

Keyword: FOR
Type: Executable
Format: line number FOR (variable) = (positive integer or variable) TO (positive integer or variable) STEP (positive integer or variable)
Description: Causes execution to cycle through the designated loop in prescribed integer steps. The step value defaults to one.

Keyword: GOSUB
Type: Executable
Format: line number GOSUB (line number)
Description: Place subroutines at the end of the program after STOP and before DATA and END. When execution encounters GOSUB, control is transferred to the subroutine labeled with the indicated line number.

Keyword: GOTO
Type: Executable
Format: line number GOTO (line number)
Description: Unconditional branch. Causes an immediate jump to the specified line. The jump may be forward or backward.

Keyword: IF
Type: Executable
Format: line number IF (condition) { THEN (statement) / THEN (line number) / GOTO (line number) }
Description: Conditional statement. Directs the order of program execution depending upon the truth of some mathematical relation. The condition is tested. If false, execution continues with the next line number following the IF statement. If the condition is true, the statement following the THEN statement is executed or control is transferred to

the line number given after THEN or GOTO. The deciding condition is a simple relational expression in which two mathematical expressions are separated by a relational operator. The hierarchy of operations observed during the evaluation of the condition is the same as that for assignment statements.

Keyword: LET
Type: Arithmetic
Format: line number LET (variable) = (expression)
Description: Assigns a numeric value to a variable. The LET statement performs the calculations within the expression and assigns the numeric value to the indicated variable.

Keyword: NEXT
Type: Executable
Format: line number NEXT (variable)
Description: Together, the FOR and NEXT statements specify the boundaries of the program loop. The variable following NEXT is the same variable immediately following the associated FOR. When execution encounters the NEXT statement, the computer adds the STEP expression value to the variable and checks to see if the variable is still less than or equal to the terminal expression value. When the variable exceeds the terminal expression value, control falls through the loop to the statement following the NEXT statement.

Keyword: RDIN
Type: Executable
Format: line number RDIN (variable) from (hex address)
Description: Reads the information available at the input device residing at the specified hex address and assigns it to the variable.

Keyword: READ
Type: Executable
Format: line number READ (variable, variable, . . .)
Description: Read is used to assign to the listed variables those values which are obtained from the DATA statement. READ causes the variables listed to be assigned sequential values in the collection of DATA statements. Each time READ is encountered, the next value is assigned.

Keyword: REM
Type: Non-executable
Format: line number REM (comment)
Description: Provides a method of inserting notes and messages into the program source listing. The message can contain any characters on the keyboard. BASIC ignores anything on a line following the letters REM. In this version, the line number may not be used as a target for branching.

Keyword: RESTORE
Type: Executable
Format: line number RESTORE
Description: Causes the next READ statement to begin reading data from the first DATA statement in the program, regardless of where the last data value was found.

Keyword: RETURN
Type: Executable
Format: line number RETURN
Description: Used to exit a subroutine. Returns control to the line in the main program following the one containing the calling GOSUB.

Keyword: STOP
Type: Executable
Format: line number STOP
Description: Typically used to separate subroutines from the main program. The STOP statement is equivalent to GOTO END.

Keyword: WRTOUT
Type: Executable
Format: line number WRTOUT (variable) TO (hex address)
Description: Transfers the data in the variable to the output device residing at the microprocessor memory location given by the hex address.

Each BASIC program line is preceded by a line number. Only one statement per line is allowed, and each line has a maximum length of 72 characters.

General Rules of Syntax

Line Numbers

Each program line is preceded by a line number. Line numbers:

1. start in column 1;
2. are all numeric and range from 1 to 255;
3. serve as targets for branch statements.

This version of BASIC does not sort program lines into ascending order by line number prior to compilation.

Expressions

Expressions are combinations of numbers, variables, or functions in which the innermost parenthetical quantity is evaluated first, in the following order, left to right:

1. @ (exponentiation)
2. * or / (multiplication and division)
3. + or - (addition and subtraction)

Numbers

In this version of BASIC, only positive whole numbers through 255 may be used in calculations. Similarly, only positive whole numbers are returned as the results of calculations. This is attributable to the eight bit data bus of the microprocessor, which can handle numbers only up to and including FF_{16} at one time. It is most certainly possible to accommodate values beyond this range with the microprocessor, but such an undertaking is beyond the scope of this paper.

Variables

Variables are written as single letters, or as single letters followed by single digits. For example, A, Z3, and X7 are all legal variable names.

Hex Addresses

Hexadecimal numbers are used in this version of BASIC to specify microprocessor memory locations. Hexadecimal numbers are:

1. exactly four characters long;
2. any combination of "0123456789ABCDEF";
3. not permitted to contain radices.

Mathematical Operators, Relational Symbols, and Functions

The mathematical operators, relational operators, and functions available with this modified version of BASIC are as follows:

Operator	Example	Meaning
+	A+B	Add B to A.
-	A-B	Subtract B from A.
*	A*B	Multiply A by B.
/	A/B	Divide A by B.
@	A@B	Calculate A to the B power.
=	A=B	A is equal to B.
>	A>B	A is greater than B.
<	A<B	A is less than B.
NEQ	A NEQ B	A is not equal to B.
GEQ	A GEQ B	A is greater than or equal to B.
LEQ	A LEQ B	A is less than or equal to B.
SQR	SQR(A)	Calculate square root of A.

RUNNING THE COMPILER UNDER CMS

The compiler presented in this paper is designed to be run under IBM's Conversational Monitor System (CMS) [5]. Once the compiler has been loaded onto the user's disk (two cylinders required), it can be used to translate a BASIC program into M6800 code.

First the user must create a CMS file containing the BASIC source program. The compiler reads the source code from a file called SOURCE DATA. If several BASIC programs will be kept on a disk, it may be convenient to create the source program under another file name and file type, then copy it into SOURCE DATA before performing the compile.

Next, the user's virtual storage must be set to 512K, if it is not already. This is accomplished with the CP command DEFINE STORAGE AS 512K.

The program is now ready to run in the normal manner. The CMS EXEC procedure of figure 5 is helpful if the compiler is to be run many times.

USING THE PRINTED OUTPUT TO PROGRAM A MICROPROCESSOR

The first section of the printed report generated by the compiler is a sequential listing of the M6800 program steps that are equivalent to the program entered as BASIC source code.

The user simply loads the values listed in the "HEX CONTENTS" column into the microprocessor memory locations

```
FILE: RUNPRT   EXEC     A          YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

GLOBAL TXTLIB PLIOLIB
FILEDEF SOURCE DISK SOURCE DATA A (RECFM F LRECL 80 BLOCK 80
FILEDEF TRMINP DISK TRMINP DATA A (RECFM F LRECL 80 BLOCK 80
FILEDEF KEYINP DISK KEYINP DATA A (RECFM F LRECL 80 BLOCK 80
FILEDEF DATAOUT PRINTER
FILEDEF MATRIX DISK MATRIX DATA A (RECFM F LRECL 80 BLOCK 80
LOAD CONTROL (NODUP
```

Fig. 5. CMS EXEC Procedure RUNPRT EXEC.

listed in the "HEX ADDRESS" column. Execution is initiated from address 0100, as noted at the end of the program listing.

ERROR MESSAGES

Any errors encountered in the source program during compilation result in the suppression of hex code generation and a listing of errors along with the numbers of the BASIC program lines in which they were discovered. The meanings of the error messages are self-explanatory.

They are divided here into logical groups according to the BASIC statements which produce them.

General Errors

ILLEGAL CHARACTER GROUP
ILLEGAL KEYWORD FOLLOWS LINE NUMBER
LAST PROGRAM LINE MUST BE END STATEMENT
NUMERIC NOT FOUND IN COLUMN ONE
LINE NUMBER EXCEEDS FIVE DIGITS
CHARACTERS FOUND BEYOND COLUMN 72
LINE DOES NOT START WITH A VALID LINE NUMBER

LET

IDENTIFIER MUST FOLLOW LET
EQUAL SIGN IS NOT IN PROPER POSITION
PARENTHESES MUST ENCLOSE FUNCTION ARGUMENT
INVALID SYNTAX

RDIN

VARIABLE NAME MUST FOLLOW RDIN
FROM MUST FOLLOW VARIABLE NAME
HEX ADDRESS MUST FOLLOW FROM
MULTIPLE ENTRIES FOR HEX ADDRESS

IF

RELATIONAL OPERATOR MUST FOLLOW FIRST CONDITION
INVALID ACTION SPECIFIED
THEN OR GOTO MUST FOLLOW CONDITION
INVALID COMMAND FOLLOWS CONDITION
PARENTHESES MUST ENCLOSE FUNCTION ARGUMENT
INVALID SYNTAX
UNEQUAL NUMBER OF LEFT AND RIGHT PARENTHESES

DATA

DATA ENTRIES MUST BE NUMERIC
COMMAS REQUIRED BETWEEN DATA VALUES
DATA ENTRIES EXCEED CAPACITY OF 100

RETURN

CHARACTERS APPEAR AFTER RETURN STATEMENT

STOP

CHARACTERS APPEAR AFTER STOP STATEMENT

WRTOUT

VARIABLE NAME MUST FOLLOW WRTOUT
TO MUST FOLLOW VARIABLE NAME
HEX ADDRESS MUST FOLLOW TO
MULTIPLE ENTRIES FOR HEX ADDRESS

GOTO

LINE NUMBER MUST FOLLOW GOTO STATEMENT
MULTIPLE ARGUMENTS IN GOTO STATEMENT

RESTORE

CHARACTERS APPEAR AFTER RESTORE STATEMENT

READ

READ ARGUMENT MUST BE A VARIABLE NAME
COMMAS REQUIRED BETWEEN READ ARGUMENTS

END

CHARACTERS APPEAR AFTER END STATEMENT

FOR

IDENTIFIER MUST FOLLOW FOR
EQUAL SIGN IS NOT IN PROPER POSITION
POSITIVE INTEGER OR VARIABLE MUST PRECEDE TO
LITERAL PRECEDING TO IS NOT INTEGER
TO IS MISPLACED OR MISSING
MISSING NEXT STATEMENT
POSITIVE INTEGER OR VARIABLE MUST FOLLOW TO
LITERAL FOLLOWING TO IS NOT INTEGER
ONLY STEP MAY FOLLOW INTEGER AFTER TO
POSITIVE INTEGER MUST FOLLOW STEP
LITERAL FOLLOWING STEP IS NOT INTEGER
EXTRANEOUS CHARACTER AT END OF LINE

NEXT

NEXT MUST BE FOLLOWED BY A VARIABLE
IMPROPER FOR-NEXT PAIR
CHARACTERS APPEAR AFTER VARIABLE NAME

GOSUB

LINE NUMBER MUST FOLLOW GOSUB STATEMENT
MULTIPLE ARGUMENTS IN GOSUB STATEMENT

APPENDIX B

Compiler PL/I Listings

FILE: CONTROL PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
                                                                        CON00010
/*YSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSU            CON00020
YSUYSU                                                          YSUYSU  CON00030
YSUYSU     MASTER'S THESIS  FRED ESENWEIN  1980                YSUYSU  CON00040
YSUYSU                                                          YSUYSU  CON00050
YSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSUYSU*/            CON00060
                                                                        CON00070
CONTROL : PROCEDURE OPTIONS (MAIN);                                     CON00080
                                                                        CON00090
   /* TABLES */                                                         CON00100
                                                                        CON00110
   DCL  1   ERR_TBL                  (75)      EXT,                     CON00120
            2   ERROR                CHAR      (40),                    CON00130
            2   SRS_LIN              FIXED     (3);                     CON00140
   DCL  1   PRS_TBL                  (999)     EXT,                     CON00150
            2   TOKEN                CHAR      (10),                    CON00160
            2   CARD                 FIXED     (4);                     CON00170
   DCL  1   SRS_TBL                  (500)     EXT,                     CON00180
            2   COMMAND              CHAR      (72);                    CON00190
   DCL  1   UST                      (999)     EXT,                     CON00200
            2   US_NTRY              CHAR      (3),                     CON00210
            2   UST_PTR              FIXED     (4);                     CON00220
   DCL  1   KEY_TBL                  (30)      EXT,                     CON00230
            2   KY_NTRY              CHAR      (10);                    CON00240
   DCL  1   TRM_TBL                  (30)      EXT,                     CON00250
            2   TM_NTRY              CHAR      (3),                     CON00260
            2   TRM_PRI              FIXED     (1);                     CON00270
   DCL  1   LIT_TBL                  (500)     EXT,                     CON00280
            2   LT_NTRY              CHAR      (9);                     CON00290
   DCL  1   IDN_TBL                  (99)      EXT,                     CON00300
            2   ID_NTRY              CHAR      (2);                     CON00310
   DCL  1   DTA_TBL                  (101)     EXT,                     CON00320
            2   DT_PTR               FIXED     (3);                     CON00330
   DCL  1   LBL_AD                   (500)     EXT,                     CON00340
            2   LBL_PTR              FIXED     (4),                     CON00350
            2   LBL_ADR              CHAR      (4);                     CON00360
   DCL  1   TMP_AD                   (70)      EXT,                     CON00370
            2   TMP_ADR              CHAR      (4);                     CON00380
   DCL  1   LIT_AD                   (500)     EXT,                     CON00390
            2   LIT_ADR              CHAR      (4);                     CON00400
   DCL  1   IDN_AD                   (99)      EXT,                     CON00410
            2   IDN_ADR              CHAR      (4);                     CON00420
   DCL  1   CNTRL                    (100)     EXT,                     CON00430
            2   LAB_PTR              FIXED     (4),                     CON00440
            2   GO_ADR               FIXED     (5),                     CON00450
            2   BRANCH               CHAR      (5),                     CON00460
            2   DESTN                FIXED     (4),                     CON00470
            2   OFFSET               FIXED     (4);                     CON00480
                                                                        CON00490
   /* FILES */                                                          CON00500
                                                                        CON00510
   DCL  SOURCE    FILE    RECORD   INPUT,                               CON00520
        DATAOUT   STREAM  PRINT,                                        CON00530
        TRMINP    FILE    RECORD   INPUT,                               CON00540
        MATRIX    FILE    RECORD;                                       CON00550
```

FILE: CONTROL PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
        KEYINP    FILE       RECORD     INPUT:                          CON00560
                                                                         CON00570
    /* ITEMS */                                                         CON00580
                                                                         CON00590
DCL SUB                                FIXED      (3)  EXT.            CON00600
    SUB2                               FIXED      (3)  INIT(1),        CON00610
    SUB3                               FIXED      (3),                 CON00620
    TRMSYM                             CHAR       (80),                CON00630
    KEYWORD                            CHAR       (80),                CON00640
    LST_IDX                            FIXED      (4)  EXT.            CON00650
    TKN_CNT                            FIXED      (3),                 CON00660
    K                                  FIXED      (4),                 CON00670
    TKN_END                            FIXED      (4),                 CON00680
    CNT                                FIXED      (2)  EXT.            CON00690
    SRS_CNT                            FIXED      (3),                 CON00700
    OT_IDX                             FIXED      (3)  EXT.            CON00710
    IGT_PTR                            FIXED      (4)  EXT.            CON00720
    CRD_CTR                            FIXED      (3)  EXT.            CON00730
    MAXTEMP                            FIXED      (3)  INIT(0),        CON00740
    TMP_STR                            FIXED      (3)  EXT.            CON00750
    LIN_PTR                            FIXED      (4)  EXT.            CON00760
    OPR                                CHAR       (3)  EXT.            CON00770
    OPR_PTR                            FIXED      (4)  EXT.            CON00780
    OP1                                CHAR       (3)  EXT.            CON00790
    OP1_PTR                            FIXED      (4)  EXT.            CON00800
    OP2                                CHAR       (3)  EXT.            CON00810
    OP2_PTR                            FIXED      (4)  EXT.            CON00820
    HEXADR                             CHAR       (4)  EXT.            CON00830
    HEXCON                             CHAR       (2)  EXT.            CON00840
    HOCON                              CHAR       (9)  EXT.            CON00850
    CMTS                               CHAR       (45) EXT.            CON00860
    PGCNT                              FIXED      (2)  EXT.            CON00870
    GT_IDX                             FIXED      (4)  EXT.            CON00880
    LBL_IDX                            FIXED      (4)  EXT.            CON00890
    MEMORY_CNT                         FIXED      (5),                 CON00900
    DECADR                             FIXED      (5)  EXT.            CON00910
    TMP_IDX                            FIXED      (2)  EXT.            CON00920
    LINE                               CHAR       (80) EXT;            CON00930
                                                                         CON00940
    /* SUBROUTINES */                                                   CON00950
                                                                         CON00960
DCL PARSE2     ENTRY;                                                    CON00970
DCL ERRCHK     ENTRY;                                                    CON00980
DCL KEYREC     ENTRY     (FIXED(4));                                     CON00990
DCL IDNREC     ENTRY     (FIXED(4));                                     CON01000
DCL TRMREC     ENTRY     (FIXED(4));                                     CON01010
DCL LITREC     ENTRY     (FIXED(4));                                     CON01020
DCL REMARK     ENTRY     (FIXED(4));                                     CON01030
DCL HEXREC     ENTRY     (FIXED(4));                                     CON01040
DCL IGTCHK     ENTRY;                                                    CON01050
DCL LFTAR      ENTRY;                                                    CON01060
DCL GOTOAR     ENTRY;                                                    CON01070
DCL IFAR       ENTRY;                                                    CON01080
DCL FORAR      ENTRY;                                                    CON01090
DCL NEXTAR     ENTRY;                                                    CON01100
```

FILE: CONTROL PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
DCL GOSUBAR ENTRY:                                          CON01110
DCL RETRNAR ENTRY:                                          CON01120
DCL DATAAR ENTRY:                                           CON01130
DCL READAR ENTRY:                                           CON01140
DCL RESTRAR ENTRY:                                          CON01150
DCL STOPAR ENTRY:                                           CON01160
DCL ENDAR ENTRY:                                            CON01170
DCL WTOUTAR ENTRY:                                          CON01180
DCL RDINAR ENTRY:                                           CON01190
DCL RITEOUT ENTRY:                                          CON01200
DCL HEADING ENTRY:                                          CON01210
DCL DCHXCO ENTRY:                                           CON01220
DCL DCHXCO2 ENTRY:                                          CON01230
DCL READCG ENTRY:                                           CON01240
DCL RESTRCG ENTRY:                                          CON01250
DCL RTRNCG ENTRY:                                           CON01260
DCL RDINCG ENTRY:                                           CON01270
DCL WTOUTCG ENTRY:                                          CON01280
DCL PLUSCG ENTRY:                                           CON01290
DCL MINUSCG ENTRY:                                          CON01300
DCL MTPLYCG ENTRY:                                          CON01310
DCL DIVIDCG ENTRY:                                          CON01320
DCL EQUALCG ENTRY:                                          CON01330
DCL GTCG ENTRY:                                             CON01340
DCL LTCG ENTRY:                                             CON01350
DCL LEQCG ENTRY:                                            CON01360
DCL GEQCG ENTRY:                                            CON01370
DCL NEQCG ENTRY:                                            CON01380
DCL EXPCG ENTRY:                                            CON01390
DCL STOPCG ENTRY:                                           CON01400
DCL ENDCG ENTRY:                                            CON01410
DCL GOTOCG ENTRY:                                           CON01420
DCL GOSUBCG ENTRY:                                          CON01430
DCL CONEOCG ENTRY:                                          CON01440
DCL SQRCG ENTRY:                                            CON01450
DCL BRANCHG ENTRY:                                          CON01460
                                                            CON01470
 /* INITIALIZE */                                          CON01480
                                                            CON01490
    SUB=1:                                                  CON01500
    CRD_CTR = 0:                                            CON01510
    CNT = 1:                                                CON01520
    LST_IDX = 1:                                            CON01530
                                                            CON01540
 /* FILL ALL TABLES WITH INERT DATA */                      CON01550
                                                            CON01560
    DO K = 1 TO 75 BY 1:                                    CON01570
        ERROR(K) = ' ':                                     CON01580
        SRS_LIN(K) = 0:                                     CON01590
    END:                                                    CON01600
                                                            CON01610
    DO K = 1 TO 999 BY 1:                                   CON01620
        TOKEN(K) = ' ':                                     CON01630
    END:                                                    CON01640
                                                            CON01650
```

```
      DO K = 1 TO 999 BY 1:                                          CON01660
         US_NTRY(K) = ' ':                                           CON01670
         UST_PTR(K) = 0:                                             CON01680
      END:                                                           CON01690
                                                                     CON01700
      DO K = 1 TO 30 BY 1:                                           CON01710
         KY_NTRY(K) = ' ':                                           CON01720
      END:                                                           CON01730
                                                                     CON01740
      DO K = 1 TO 30 BY 1:                                           CON01750
         TM_NTRY(K) = ' ':                                           CON01760
         TRM_PRI(K) = 9:                                             CON01770
      END:                                                           CON01780
                                                                     CON01790
      LT_NTRY(1) = '1':                                              CON01800
      LIT_ADR(1) = ' ':                                              CON01810
      LT_NTRY(2) = '99':                                             CON01820
      LIT_ADR(2) = ' ':                                              CON01830
      DO K = 3 TO 500 BY 1:                                          CON01840
         LT_NTRY(K) = ' ':                                           CON01850
         LIT_ADR(K) = ' ':                                           CON01860
      END:                                                           CON01870
                                                                     CON01880
      DO K = 1 TO 99 BY 1:                                           CON01890
         ID_NTRY(K) = ' ':                                           CON01900
         IDN_ADR(K) = ' ':                                           CON01910
      END:                                                           CON01920
                                                                     CON01930
      DO K = 1 TO 101 BY 1:                                          CON01940
         DT_PTR(K) = 0:                                              CON01950
      END:                                                           CON01960
                                                                     CON01970
      DO K = 1 TO 500 BY 1:                                          CON01980
         LBL_PTR(K) = 0:                                             CON01990
         LBL_ADR(K) = ' ':                                           CON02000
      END:                                                           CON02010
                                                                     CON02020
      DO K = 1 TO 70 BY 1:                                           CON02030
         TMP_ADR(K) = ' ':                                           CON02040
      END:                                                           CON02050
                                                                     CON02060
      DO K = 1 TO 100 BY 1:                                          CON02070
         LAB_PTR(K) = 0:                                             CON02080
         GO_ADR(K) = 0:                                              CON02090
         BRANCH(K) = ' ':                                            CON02100
         DESTN(K) = 0:                                               CON02110
         OFFSET(K) =0:                                               CON02120
      END:                                                           CON02130
                                                                     CON02140
   /* OPEN FILES */                                                  CON02150
                                                                     CON02160
      ON ENDFILE(SOURCE) GO TO T002:                                 CON02170
      ON ENDFILE(KEYINP) GO TO T001:                                 CON02180
      ON ENDFILE(TRMINP) GO TO C005:                                 CON02190
      OPEN FILE(TRMINP):                                             CON02200
```

FILE: CONTROL PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
      OPEN FILE(KEYINP);                                          CON02210
      OPEN FILE(SOURCE);                                          CON02220
      OPEN FILE (MATRIX) OUTPUT;                                  CON02230
                                                                  CON02240
/* FILL TERMINAL TABLE FROM EXTERNAL FILE */                      CON02250
                                                                  CON02260
      K = 1;                                                      CON02270
C004:READ FILE(TRMINP) INTO (TRMSYM);                             CON02280
      TM_NTRY(K) = TRMSYM;                                        CON02290
      K = K + 1;                                                  CON02300
      GO TO C004;                                                 CON02310
                                                                  CON02320
/* FILL KEYWORD TABLE FROM EXTERNAL FILE */                       CON02330
                                                                  CON02340
C005:K = 1;                                                       CON02350
C003:READ FILE(KEYINP) INTO (KEYWORD);                            CON02360
      KY_NTRY(K) = KEYWORD;                                       CON02370
      K = K + 1;                                                  CON02380
      GO TO C003;                                                 CON02390
                                                                  CON02400
      /*******************************************               CON02410
      ***********   LEXICAL ANALYSIS   ***********                CON02420
      *******************************************/               CON02430
                                                                  CON02440
/* READ SOURCE LINE */                                           CON02450
                                                                  CON02460
T001:READ FILE (SOURCE) INTO (LINE);                              CON02470
      DISPLAY (CRD_CTR);                                          CON02480
      CRD_CTR = CRD_CTR + 1;                                      CON02490
      SRS_CNT = CRD_CTR;                                          CON02500
                                                                  CON02510
/* FOR ADDITION OF LINE NUMBER TO SOURCE LISTING */               CON02520
                                                                  CON02530
      COMMAND(CRD_CTR) = SUBSTR(LINE, 1, 72);                     CON02540
                                                                  CON02550
/* IGNORE ADDITIONAL ERRORS IF MORE THAN 25 ARE FOUND */          CON02560
                                                                  CON02570
      IF CNT > 25 THEN GO TO CABND;                               CON02580
                                                                  CON02590
/* CHECK SOURCE LINE FOR ERRORS */                                CON02600
                                                                  CON02610
      CALL ERRCHK ;                                               CON02620
                                                                  CON02630
/* PARSE SOURCE LINE AND BUILD PARSE TABLE */                     CON02640
                                                                  CON02650
C001:CALL PARSE2;                                                 CON02660
      GO TO T001;                                                 CON02670
                                                                  CON02680
/* SEARCH TABLES FOR TOKEN MATCH */                               CON02690
                                                                  CON02700
T002:K = 1;                                                       CON02710
      DO WHILE (TOKEN(K) ¬= ' ');                                 CON02720
         TKN_END = LST_IDX;                                       CON02730
         IF TOKEN(K) = 'REM       ' THEN CALL REMARK(K);          CON02740
         CALL KEYREC(K);                                          CON02750
```

```
            IF UST_IDX > TKN_END THEN GO TO C006:                    CON02760
            CALL TRMREC(K):                                          CON02770
            IF UST_IDX > TKN_END THEN GO TO C006:                    CON02780
            CALL IDNREC(K):                                          CON02790
            IF UST_IDX > TKN_END THEN GO TO C006:                    CON02800
            CALL LITREC(K):                                          CON02810
            IF UST_IDX > TKN_END THEN GO TO C006:                    CON02820
            CALL HEXREC(K):                                          CON02830
            IF UST_IDX > TKN_END THEN GO TO C006:                    CON02840
            ERROR(CNT) = 'ILLEGAL CHARACTER GROUP':                  CON02850
            SRS_LIN(CNT) = CARD(K):                                  CON02860
            CNT = CNT + 1:                                           CON02870
        C006:K = K + 1:                                              CON02880
        END:                                                         CON02890
                                                                     CON02900
        /*****************************************                   CON02910
        ***********   SYNTAX ANALYSIS   ***********                  CON02920
        *****************************************/                  CON02930
                                                                     CON02940
    /* INITIALIZE */                                                 CON02950
                                                                     CON02960
        DT_IDX = 1:                                                  CON02970
        UST_IDX = 1:                                                 CON02980
        CRD_CTR = 1:                                                 CON02990
    C007:DISPLAY (CRD_CTR):                                          CON03000
        IF (US_NTRY(UST_IDX)=' ') THEN GO TO END_OF_SYNTAX:          CON03010
        SUB3 = CNT:                                                  CON03020
                                                                     CON03030
    /* IGNORE ADDITIONAL ERRORS IF MORE THAN 50 ARE FOUND */         CON03040
                                                                     CON03050
        IF CNT > 50 THEN GO TO CABND:                                CON03060
        CALL TGTCHK:                                                 CON03070
        IF SUB3¬=CNT THEN GO TO C007:                                CON03080
                                                                     CON03090
    /* LINE NUMBER HAS BEEN VERIFIED */                              CON03100
                                                                     CON03110
        IF(US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=23) THEN        CON03120
            DO:                                                      CON03130
                CRD_CTR=CRD_CTR+1:                                   CON03140
                UST_IDX=UST_IDX+1:                                   CON03150
                GO TO C007:                                          CON03160
            END:                                                     CON03170
        IF US_NTRY(UST_IDX)¬='KEY' THEN GO TO C008:                  CON03180
        IF UST_PTR(UST_IDX)=1 THEN                                   CON03190
            DO:                                                      CON03200
                CALL LFTAR:                                          CON03210
                IF TMP_STR > MAXTEMP THEN MAXTEMP = TMP_STR:         CON03220
                GO TO C007:                                          CON03230
            END:                                                     CON03240
        IF UST_PTR(UST_IDX)=2 THEN                                   CON03250
            DO:                                                      CON03260
                CALL GOTOAR:                                         CON03270
                GO TO C007:                                          CON03280
            END:                                                     CON03290
        IF UST_PTR(UST_IDX)=3 THEN                                   CON03300
```

```
            DO;                                                    CON03310
                CALL IEAR;                                         CON03320
                IF TMP_STR > MAXTEMP THEN MAXTEMP = TMP_STR;       CON03330
                IF MAXTEMP < 2 THEN MAXTEMP = 2;                   CON03340
                GO TO C007;                                        CON03350
            END;                                                   CON03360
        IF UST_PTR(UST_IDX)=6 THEN                                 CON03370
            DO;                                                    CON03380
                CALL FORAR;                                        CON03390
                GO TO C007;                                        CON03400
            END;                                                   CON03410
        IF UST_PTR(UST_IDX)=7 THEN                                 CON03420
            DO;                                                    CON03430
                CALL NEXTAR;                                       CON03440
                IF MAXTEMP = 0 THEN MAXTEMP = 1;                   CON03450
                GO TO C007;                                        CON03460
            END;                                                   CON03470
        IF UST_PTR(UST_IDX)=8 THEN                                 CON03480
            DO;                                                    CON03490
                CALL GOSUBAR;                                      CON03500
                GO TO C007;                                        CON03510
            END;                                                   CON03520
        IF UST_PTR(UST_IDX)=9 THEN                                 CON03530
            DO;                                                    CON03540
                CALL RETRNAR;                                      CON03550
                GO TO C007;                                        CON03560
            END;                                                   CON03570
        IF UST_PTR(UST_IDX)=12 THEN                                CON03580
            DO;                                                    CON03590
                CALL DATAAR;                                       CON03600
                GO TO C007;                                        CON03610
            END;                                                   CON03620
        IF UST_PTR(UST_IDX)=13 THEN                                CON03630
            DO;                                                    CON03640
                CALL READAR;                                       CON03650
                GO TO C007;                                        CON03660
            END;                                                   CON03670
        IF UST_PTR(UST_IDX)=14 THEN                                CON03680
            DO;                                                    CON03690
                CALL RESTRAR;                                      CON03700
                GO TO C007;                                        CON03710
            END;                                                   CON03720
        IF UST_PTR(UST_IDX)=16 THEN                                CON03730
            DO;                                                    CON03740
                CALL STOPAR;                                       CON03750
                GO TO C007;                                        CON03760
            END;                                                   CON03770
        IF UST_PTR(UST_IDX)=17 THEN                                CON03780
            DO;                                                    CON03790
                CALL ENDAR;                                        CON03800
                GO TO C007;                                        CON03810
            END;                                                   CON03820
        IF UST_PTR(UST_IDX)=18 THEN                                CON03830
            DO;                                                    CON03840
                CALL WTOUTAR;                                      CON03850
```

```
                GO TO C007;                                          CON03860
            END;                                                     CON03870
        IF UST_PTR(UST_IDX)=19 THEN                                  CON03880
            DO;                                                      CON03890
                CALL BOINAR;                                         CON03900
                GO TO C007;                                          CON03910
            END;                                                     CON03920
C008:ERROR(CNT)='ILLEGAL KEYWORD FOLLOWS LINE NUMBER';               CON03930
        SRS_LIN(CNT)=CRD_CTR;                                        CON03940
        CNT=CNT+1;                                                   CON03950
        DO WHILE((US_NTRY(UST_IDX)-='TRM')|(UST_PTR(UST_IDX)-=23));  CON03960
            UST_IDX = UST_IDX + 1;                                   CON03970
        END;                                                         CON03980
        UST_IDX = UST_IDX + 1;                                       CON03990
        CRD_CTR = CRD_CTR + 1;                                       CON04000
        GO TO C007;                                                  CON04010
END_OF_SYNTAX:                                                       CON04020
                                                                     CON04030
/* IS LAST SYMBOL AN END STATEMENT ? */                              CON04040
    IF (US_NTRY(UST_IDX-2) -= 'KEY')|(UST_PTR(UST_IDX-2) -= 17)      CON04050
        THEN DO;                                                     CON04060
            ERROR(CNT) = 'LAST PROGRAM LINE MUST BE END STMT';       CON04070
            SRS_LIN(CNT) = CRD_CTR;                                  CON04080
        END;                                                         CON04090
                                                                     CON04100
/* DIAGNOSTIC SUMMARY */                                            CON04110
                                                                     CON04120
    IF ERROR(1)-=' ' THEN                                            CON04130
        DO;                                                          CON04140
            CNT = CNT + 2;                                           CON04150
            ERROR(CNT)='ABEND - SEVERE ERRORS DETECTED.';            CON04160
            CNT = CNT + 1;                                           CON04170
            ERROR(CNT)='CODE GENERATION SUPPRESSED.';                CON04180
            CNT = CNT + 1;                                           CON04190
            GO TO CABND;                                             CON04200
        END;                                                         CON04210
    ERROR(1)='NO DIAGNOSTICS GENERATED';                             CON04220
    ERROR(3)='MAXIMUM TEMPORARY STORAGE USED -';                     CON04230
    SRS_LIN(3)=MAXTEMP;                                              CON04240
    CNT=4;                                                           CON04250
                                                                     CON04260
    /*****************************************                      CON04270
    **********  CODE GENERATION  **********                         CON04280
    *****************************************/                      CON04290
                                                                     CON04300
    PGCNT=0;                                                         CON04310
    DECADR=0;                                                        CON04320
                                                                     CON04330
/* HEAD FIRST OUTPUT PAGE */                                        CON04340
                                                                     CON04350
    CALL HEADING;                                                    CON04360
                                                                     CON04370
/* ENTER DATA TABLE INTO UP MEMORY */                               CON04380
                                                                     CON04390
    K = 1;                                                           CON04400
```

```
        DO WHILE (DT_PTR(K) ¬= 0):                                          CON04410
            MOCON = LT_NTRY(DT_PTR(K)):                                     CON04420
            CALL DCHXC02:                                                   CON04430
            CMTS = 'DATA':                                                  CON04440
            CALL RITEOUT:                                                   CON04450
            K=K+1:                                                          CON04460
        END:                                                                CON04470
                                                                            CON04480
    /* ENTER LITERALS INTO UP MEMORY */                                     CON04490
                                                                            CON04500
        K = 1:                                                              CON04510
        DO WHILE (LT_NTRY(K) ¬= ' '):                                       CON04520
            MOCON = LT_NTRY(K):                                             CON04530
            IF (VERIFY(MOCON,'0123456789 .')) ¬= 0 THEN GO TO C009:         CON04540
            CALL DCHXC02:                                                   CON04550
            CMTS = 'CONSTANT':                                              CON04560
            CALL RITEOUT:                                                   CON04570
            LIT_ADR(K)=HEXADR:                                              CON04580
C009:       K = K + 1:                                                      CON04590
        END:                                                                CON04600
                                                                            CON04610
    /* RESERVE UP MEMORY LOCATIONS FOR IDENTIFIERS */                       CON04620
                                                                            CON04630
        K = 1:                                                              CON04640
        DO WHILE (ID_NTRY(K) ¬= ' '):                                       CON04650
            HEXCON = 'XX':                                                  CON04660
            MOCON = 'XXXXXXXXX':                                            CON04670
            CMTS = ID_NTRY(K):                                              CON04680
            CALL RITEOUT:                                                   CON04690
            IDN_ADR (K) = HEXADR:                                           CON04700
            K = K + 1:                                                      CON04710
        END:                                                                CON04720
                                                                            CON04730
    /* RESERVE UP MEMORY LOCATIONS FOR TEMPORARY STORAGE */                 CON04740
                                                                            CON04750
        DO K=1 TO MAXTEMP BY 1:                                             CON04760
            HEXCON = 'XX':                                                  CON04770
            MOCON='XXXXXXXXX':                                              CON04780
            CMTS = 'TEMPORARY STORAGE LOCATION':                            CON04790
            CALL RITEOUT:                                                   CON04800
            TMP_ADR(K) = HEXADR:                                            CON04810
        END:                                                                CON04820
                                                                            CON04830
    /* JUMP TO HEX ADDRESS 0100 */                                          CON04840
        DECADR = 256:                                                       CON04850
                                                                            CON04860
    /* INITIALIZE INDEX REGISTER (DATA TABLE POINTER) */                    CON04870
                                                                            CON04880
        CALL RESTRCG:                                                       CON04890
                                                                            CON04900
    /* PREPARE TO READ MATRIX INFORMATION */                                CON04910
                                                                            CON04920
        GT_IDX = 1:                                                         CON04930
        K = 1:                                                              CON04940
        CLOSE FILE (MATRIX):                                                CON04950
```

```
        ON ENDFILE (MATRIX) GO TO C010:                                  CON04960
        OPEN FILE (MATRIX) INPUT:                                        CON04970
                                                                         CON04980
    /* READ A MATRIX LINE */                                             CON04990
                                                                         CON05000
READ_MATRIX:                                                             CON05010
        READ FILE (MATRIX) INTO (LINE):                                  CON05020
        LIN_PTR = SUBSTR (LINE, 1, 4):                                   CON05030
        OPR = SUBSTR (LINE, 5, 3):                                       CON05040
        OPR_PTR = SUBSTR (LINE, 8, 4):                                   CON05050
        OP1 = SUBSTR (LINE, 12, 3):                                      CON05060
        OP1_PTR = SUBSTR (LINE, 15, 4):                                  CON05070
        OP2 = SUBSTR (LINE, 19, 3):                                      CON05080
        OP2_PTR = SUBSTR (LINE, 22, 4):                                  CON05090
                                                                         CON05100
    /* IS THIS THE FIRST ENCOUNTER WITH THIS LINE NUMBER ? */            CON05110
                                                                         CON05120
        IF LIN_PTR ¬= LBL_PTR(K) THEN                                    CON05130
            DO:                                                          CON05140
                TMP_IDX = 1:                                             CON05150
                K = K + 1:                                               CON05160
                LBL_PTR(K) = LIN_PTR:                                    CON05170
                CALL DCHXCO:                                             CON05180
                LBL_ADR(K) = HEXADR:                                     CON05190
            END:                                                         CON05200
                                                                         CON05210
    /* CALL APPROPRIATE CODE GENERATION SUBROUTINE */                    CON05220
                                                                         CON05230
        /* READ */                                                       CON05240
                                                                         CON05250
        IF((OPR = 'KEY')&(OPR_PTR = 13)) THEN                            CON05260
            DO:                                                          CON05270
                CALL READCG:                                             CON05280
                GO TO READ_MATRIX:                                       CON05290
            END:                                                         CON05300
                                                                         CON05310
        /* RESTORE */                                                    CON05320
                                                                         CON05330
        IF ((OPR = 'KEY')&(OPR_PTR = 14)) THEN                           CON05340
            DO:                                                          CON05350
                CALL RESTRCG:                                            CON05360
                GO TO READ_MATRIX:                                       CON05370
            END:                                                         CON05380
                                                                         CON05390
        /* RETURN */                                                     CON05400
                                                                         CON05410
        IF ((OPR = 'KEY')&(OPR_PTR = 9)) THEN                            CON05420
            DO:                                                          CON05430
                CALL RTRNCG:                                             CON05440
                GO TO READ_MATRIX:                                       CON05450
            END:                                                         CON05460
                                                                         CON05470
        /* RDIN */                                                       CON05480
                                                                         CON05490
        IF ((OPR = 'KEY')&(OPR_PTR = 19)) THEN                           CON05500
```

```
        DO;                                                     CON05510
            CALL ROTNCG;                                        CON05520
            GO TO READ_MATRIX;                                  CON05530
        END;                                                    CON05540
                                                                CON05550
    /* WRTOUT */                                                CON05560
                                                                CON05570
    IF ((OPR = 'KEY')&(OPR_PTR = 18)) THEN                      CON05580
        DO;                                                     CON05590
            CALL WTOUTCG;                                       CON05600
            GO TO READ_MATRIX;                                  CON05610
        END;                                                    CON05620
                                                                CON05630
    /* + */                                                     CON05640
                                                                CON05650
    IF ((OPR = 'TRM')&(OPR_PTR = 2)) THEN                       CON05660
        DO;                                                     CON05670
            CALL PLUSCG;                                        CON05680
            GO TO READ_MATRIX;                                  CON05690
        END;                                                    CON05700
                                                                CON05710
    /* - */                                                     CON05720
                                                                CON05730
    IF ((OPR = 'TRM')&(OPR_PTR = 1)) THEN                       CON05740
        DO;                                                     CON05750
            CALL MINUSCG;                                       CON05760
            GO TO READ_MATRIX;                                  CON05770
        END;                                                    CON05780
                                                                CON05790
    /* * */                                                     CON05800
                                                                CON05810
    IF ((OPR = 'TRM')&(OPR_PTR = 4)) THEN                       CON05820
        DO;                                                     CON05830
            CALL MTPLYCG;                                       CON05840
            GO TO READ_MATRIX;                                  CON05850
        END;                                                    CON05860
                                                                CON05870
    /* / */                                                     CON05880
                                                                CON05890
    IF ((OPR = 'TRM')&(OPR_PTR = 3)) THEN                       CON05900
        DO;                                                     CON05910
            CALL DIVIDCG;                                       CON05920
            GO TO READ_MATRIX;                                  CON05930
        END;                                                    CON05940
                                                                CON05950
    /* = */                                                     CON05960
                                                                CON05970
    IF ((OPR = 'TRM')&(OPR_PTR = 6)) THEN                       CON05980
        DO;                                                     CON05990
            CALL EQUALCG;                                       CON06000
            GO TO READ_MATRIX;                                  CON06010
        END;                                                    CON06020
                                                                CON06030
    /* CONDITIONAL = */                                         CON06040
                                                                CON06050
```

```
        IF ((OPR = 'TRM')&(OPR_PTR = 27)) THEN                    CON06060
             DO:                                                  CON06070
                  CALL CONFOCG:                                   CON06080
                  GO TO READ_MATRIX:                              CON06090
             END:                                                 CON06100
                                                                  CON06110
        /* > */                                                   CON06120
                                                                  CON06130
        IF ((OPR = 'TRM')&(OPR_PTR = 9)) THEN                     CON06140
             DO:                                                  CON06150
                  CALL GTCG:                                      CON06160
                  GO TO READ_MATRIX:                              CON06170
             END:                                                 CON06180
                                                                  CON06190
        /* < */                                                   CON06200
                                                                  CON06210
        IF ((OPR = 'TRM')&(OPR_PTR = 10)) THEN                    CON06220
             DO:                                                  CON06230
                  CALL LTCG:                                      CON06240
                  GO TO READ_MATRIX:                              CON06250
             END:                                                 CON06260
                                                                  CON06270
        /* LEQ */                                                 CON06280
                                                                  CON06290
        IF ((OPR = 'TRM')&(OPR_PTR = 24)) THEN                    CON06300
             DO:                                                  CON06310
                  CALL LEQCG:                                     CON06320
                  GO TO READ_MATRIX:                              CON06330
             END:                                                 CON06340
                                                                  CON06350
        /* GEQ */                                                 CON06360
                                                                  CON06370
        IF ((OPR = 'TRM')&(OPR_PTR = 25)) THEN                    CON06380
             DO:                                                  CON06390
                  CALL GEQCG:                                     CON06400
                  GO TO READ_MATRIX:                              CON06410
             END:                                                 CON06420
                                                                  CON06430
        /* NEQ */                                                 CON06440
                                                                  CON06450
        IF ((OPR = 'TRM')&(OPR_PTR = 26)) THEN                    CON06460
             DO:                                                  CON06470
                  CALL NEQCG:                                     CON06480
                  GO TO READ_MATRIX:                              CON06490
             END:                                                 CON06500
                                                                  CON06510
        /* SQUARE ROOT */                                         CON06520
                                                                  CON06530
        IF ((OPR = 'TRM')&(OPR_PTR = 21)) THEN                    CON06540
             DO:                                                  CON06550
                  CALL SQRCG:                                     CON06560
                  GO TO READ_MATRIX:                              CON06570
             END:                                                 CON06580
                                                                  CON06590
        /* R */                                                   CON06600
```

```
                                                                        CON06610
      IF ((OPR = 'TRM')&(OPR_PTR = 5)) THEN                             CON06620
         DO:                                                            CON06630
            CALL EXPCG;                                                 CON06640
            GO TO READ_MATRIX;                                          CON06650
         END;                                                           CON06660
                                                                        CON06670
      /* STOP */                                                        CON06680
                                                                        CON06690
      IF ((OPR = 'KEY')&(OPR_PTR = 16)) THEN                            CON06700
         DO:                                                            CON06710
            CALL STOPCG;                                                CON06720
            GO TO READ_MATRIX;                                          CON06730
         END;                                                           CON06740
                                                                        CON06750
      /* END */                                                         CON06760
                                                                        CON06770
      IF ((OPR = 'KEY')&(OPR_PTR = 17)) THEN                            CON06780
         DO:                                                            CON06790
            CALL ENDCG;                                                 CON06800
            GO TO READ_MATRIX;                                          CON06810
         END;                                                           CON06820
                                                                        CON06830
      /* GOTO */                                                        CON06840
                                                                        CON06850
      IF ((OPR = 'KEY')&(OPR_PTR = 2)) THEN                             CON06860
         DO:                                                            CON06870
            CALL GOTOCG;                                                CON06880
            GO TO READ_MATRIX;                                          CON06890
         END;                                                           CON06900
                                                                        CON06910
      /* GOSUB */                                                       CON06920
                                                                        CON06930
      IF ((OPR = 'KEY')&(OPR_PTR = 8)) THEN                             CON06940
         DO:                                                            CON06950
            CALL GOSUBCG;                                               CON06960
            GO TO READ_MATRIX;                                          CON06970
         END;                                                           CON06980
                                                                        CON06990
      GO TO READ_MATRIX;                                                CON07000
                                                                        CON07010
C010:                                                                   CON07020
      MEMORY_CNT = DECADR;                                              CON07030
                                                                        CON07040
   /* COMPLETE COMPILATION BY ASSIGNING DESTINATION ADDRESSES TO        CON07050
      BRANCH STATEMENTS */                                              CON07060
                                                                        CON07070
      CALL BRANCHG;                                                     CON07080
                                                                        CON07090
   /* WRITE TOTAL UP MEMORY REQUIRED */                                 CON07100
                                                                        CON07110
      PUT FILE (DATAOUT) SKIP;                                          CON07120
      PUT FILE (DATAOUT) SKIP EDIT ('PROGRAM STARTING ADDRESS',         CON07130
         '0100') (X(5), A(24), X(1), A(4));                             CON07140
      PUT FILE (DATAOUT) SKIP;                                          CON07150
```

FILE: CONTROL PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
        PUT FILE (DATAOUT) SKIP EDIT ('TOTAL MEMORY USED :',          CON07160
         MEMORY_CNT, 'BYTES') (X(5), A(19), X(2), F(5), X(1), A(5));  CON07170
                                                                      CON07180
        /*****************************************************       CON07190
        ************   SEND REPORTS TO PRINTER   ************       CON07200
        *****************************************************/      CON07210
                                                                      CON07220
    /* WRITE SOURCE LISTING WITH CARD NUMBER */                       CON07230
                                                                      CON07240
  CABND : PUT FILE (DATAOUT) PAGE;                                    CON07250
        PUT FILE(DATAOUT) SKIP EDIT ('SOURCE LISTING')                CON07260
            (X(11), A(14));                                           CON07270
        PUT FILE(DATAOUT) SKIP;                                       CON07280
        DO WHILE (SUB2 <= SRS_CNT);                                   CON07290
            PUT FILE(DATAOUT) SKIP EDIT (COMMAND(SUB2), SUB2)         CON07300
                (X(3), A(72), F(3));                                  CON07310
            SUB2 = SUB2 + 1;                                          CON07320
        END;                                                          CON07330
                                                                      CON07340
    /* WRITE ERROR TABLE */                                           CON07350
                                                                      CON07360
        TKN_CNT = CNT;                                                CON07370
        CNT = 1;                                                      CON07380
        PUT FILE(DATAOUT) PAGE;                                       CON07390
        PUT FILE(DATAOUT) SKIP EDIT ('ERROR TABLE')                   CON07400
            (X(11), A(11));                                           CON07410
        PUT FILE(DATAOUT) SKIP;                                       CON07420
        PUT FILE(DATAOUT) SKIP EDIT ('ERROR', 'SOURCE LINE')          CON07430
            (X(3), A(5), X(35), A(11));                               CON07440
        PUT FILE(DATAOUT) SKIP;                                       CON07450
        DO WHILE (CNT < TKN_CNT);                                     CON07460
        PUT FILE(DATAOUT) SKIP EDIT (ERROR(CNT), SRS_LIN(CNT))        CON07470
            (X(3), A(40), F(3));                                      CON07480
        CNT = CNT + 1;                                                CON07490
        END;                                                          CON07500
                                                                      CON07510
                                                                      CON07520
    /* WRITE KEYWORD TABLE */                                         CON07530
                                                                      CON07540
        PUT FILE (DATAOUT) PAGE;                                      CON07550
        PUT FILE (DATAOUT) SKIP EDIT ('KEYWORD TABLE')                CON07560
            (X(11), A(13));                                           CON07570
        PUT FILE (DATAOUT) SKIP;                                      CON07580
        SUB = 1;                                                      CON07590
        DO WHILE (KY_NTRY(SUB) ^= ' ');                               CON07600
            PUT FILE (DATAOUT) SKIP EDIT (KY_NTRY(SUB), SUB)          CON07610
                (X(3), A(10), X(3), F(3));                            CON07620
            SUB = SUB + 1;                                            CON07630
        END;                                                          CON07640
                                                                      CON07650
    /* WRITE TERMINAL TABLE */                                        CON07660
                                                                      CON07670
        PUT FILE (DATAOUT) PAGE;                                      CON07680
        PUT FILE (DATAOUT) SKIP EDIT ('TERMINAL TABLE')               CON07690
            (X(11), A(14));                                           CON07700
```

FILE: CONTROL PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
        PUT FILE (DATAOUT) SKIP;                                        CON07710
        SUB = 1;                                                        CON07720
        DO WHILE (TM_NTRY(SUB) ¬= ' ');                                 CON07730
             PUT FILE (DATAOUT) SKIP EDIT (TM_NTRY(SUB), SUB)           CON07740
                  (X(3), A(3), X(3), F(3));                             CON07750
             SUB = SUB + 1;                                             CON07760
        END;                                                            CON07770
                                                                        CON07780
    /* WRITE IDENTIFIER TABLE */                                        CON07790
                                                                        CON07800
        PUT FILE (DATAOUT) PAGE;                                        CON07810
        PUT FILE (DATAOUT) SKIP EDIT ('IDENTIFIER TABLE')               CON07820
             (X(11), A(16));                                            CON07830
        PUT FILE (DATAOUT) SKIP;                                        CON07840
        SUB = 1;                                                        CON07850
        DO WHILE (ID_NTRY(SUB) ¬= ' ');                                 CON07860
             PUT FILE (DATAOUT) SKIP EDIT (ID_NTRY(SUB), SUB)           CON07870
                  (X(3), A(2), X(3), F(3));                             CON07880
             SUB = SUB + 1;                                             CON07890
        END;                                                            CON07900
                                                                        CON07910
    /* WRITE LITERAL TABLE */                                           CON07920
                                                                        CON07930
        PUT FILE (DATAOUT) PAGE;                                        CON07940
        PUT FILE (DATAOUT) SKIP EDIT ('LITERAL TABLE')                  CON07950
             (X(11), A(13));                                            CON07960
        PUT FILE (DATAOUT) SKIP;                                        CON07970
        SUB = 1;                                                        CON07980
        DO WHILE (LT_NTRY(SUB) ¬= ' ');                                 CON07990
             PUT FILE (DATAOUT) SKIP EDIT (LT_NTRY(SUB), SUB)           CON08000
                  (X(3), A(9), X(3), F(3));                             CON08010
             SUB = SUB + 1;                                             CON08020
        END;                                                            CON08030
                                                                        CON08040
    /* WRITE UNIFORM SYMBOL TABLE */                                    CON08050
                                                                        CON08060
        PUT FILE(DATAOUT) PAGE;                                         CON08070
        PUT FILE(DATAOUT) SKIP EDIT ('UNIFORM SYMBOL TABLE')            CON08080
             (X(11), A(20));                                            CON08090
        PUT FILE(DATAOUT) SKIP;                                         CON08100
        PUT FILE(DATAOUT) SKIP EDIT ('SYMBOL', 'POINTER')               CON08110
             (X(3), A(6), X(3), A(7));                                  CON08120
        SUB = 1;                                                        CON08130
        DO WHILE(SUB < UST_IDX);                                        CON08140
             PUT FILE(DATAOUT) SKIP EDIT (US_NTRY(SUB), UST_PTR(SUB))   CON08150
                  (X(3), A(3), X(6), F(4));                             CON08160
             SUB = SUB + 1;                                             CON08170
        END;                                                            CON08180
                                                                        CON08190
    /* WRITE DATA TABLE */                                              CON08200
                                                                        CON08210
        PUT FILE (DATAOUT) PAGE;                                        CON08220
        PUT FILE (DATAOUT) SKIP EDIT ('DATA TABLE')                     CON08230
             (X(11),A(10));                                             CON08240
        PUT FILE (DATAOUT) SKIP;                                        CON08250
```

FILE: CONTROL PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
          PUT FILE (DATAOUT) SKIP EDIT ('LIT TBL POINTER')          CONOR260
               (X(3),A(15));                                        CONOR270
          SUB = 1;                                                  CONOR280
          DO WHILE (DT_PTR(SUB)¬=0);                                CONOR290
               PUT FILE (DATAOUT) SKIP EDIT (DT_PTR(SUB))           CONOR300
                    (X(3),F(3));                                    CONOR310
               SUB = SUB + 1;                                       CONOR320
          END;                                                      CONOR330
                                                                    CONOR340
     /* CLOSE FILES */                                            CONOR350
                                                                    CONOR360
          CLOSE FILE(SOURCE);                                       CONOR370
          CLOSE FILE (MATRIX);                                      CONOR380
          CLOSE FILE (KEYINP);                                      CONOR390
          CLOSE FILE(TRMINP);                                       CONOR400
END CONTROL;                                                        CONOR410
```

```
FILE: PARSE2    OPTIONS    A          YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

PARSE2 : PROCEDURE:                                                    PAR00010
                                                                       PAR00020
DCL  1   PRS_TAB                    (999)     EXT.                     PAR00030
         2   TOKEN                  CHAR      (10).                    PAR00040
         2   CARD                   FIXED     (4);                     PAR00050
                                                                       PAR00060
DCL  SUB                            FIXED     (3)  EXT.                PAR00070
     N                              FIXED     (2).                     PAR00080
     P                              FIXED     (2).                     PAR00090
     LINE                           CHAR      (80)  EXT.               PAR00100
     CRD_CTR                        FIXED     (3)  EXT.                PAR00110
     BREAK                          CHAR      (80).                    PAR00120
     ARK_CHR                        CHAR      (12)                     PAR00130
          INIT(' -+/*@=()><.');                                        PAR00140
     ALF_CHR                        CHAR      (37)                     PAR00150
          INIT(' ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.');              PAR00160
                                                                       PAR00170
/* CHECK FOR LEADING BLANKS */                                         PAR00180
                                                                       PAR00190
     N = VERIFY (LINE, ARK_CHR);                                       PAR00200
     IF N=0 THEN N=73;                                                 PAR00210
     LINE = SUBSTR(LINE, N);                                           PAR00220
                                                                       PAR00230
     DO WHILE (LINE ^= ' ');                                           PAR00240
          N = VERIFY (LINE, ALF_CHR);                                  PAR00250
          TOKEN(SUB) = SUBSTR (LINE, 1, N-1);                          PAR00260
          CARD(SUB) = CRD_CTR;                                         PAR00270
          SUB = SUB + 1;                                               PAR00280
          LINE = SUBSTR (LINE, N);                                     PAR00290
          N = VERIFY (LINE, ARK_CHR);                                  PAR00300
          IF N = 0 THEN N = 73;                                        PAR00310
          BREAK = SUBSTR (LINE, 1, N-1);                               PAR00320
T005:     P = VERIFY (BREAK, ' ');                                     PAR00330
          IF P = 0 THEN GO TO T004;                                    PAR00340
          TOKEN(SUB) = SUBSTR (BREAK, P, 1);                           PAR00350
          CARD(SUB) = CRD_CTR;                                         PAR00360
          SUB = SUB + 1;                                               PAR00370
          BREAK = SUBSTR (BREAK, P + 1);                               PAR00380
          GO TO T005;                                                  PAR00390
T004:     LINE = SUBSTR (LINE, N);                                     PAR00400
     END;                                                              PAR00410
                                                                       PAR00420
     TOKEN(SUB) = '$          ';                                       PAR00430
     CARD(SUB) = CRD_CTR;                                              PAR00440
     SUB = SUB + 1;                                                    PAR00450
     RETURN;                                                           PAR00460
END PARSE2;                                                            PAR00470
```

```
FILE: RCHK   PLIOPT   A          YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

ERRCHK: PROCEDURE:                                                    ERR00010
                                                                      ERR00020
        1  ERR_TAB                  (25)      EXT.                   ERR00030
           2  ERROR                 CHAR(40).                        ERR00040
           2  SRS_LIN               FIXED(3):                        ERR00050
                                                                      ERR00060
        LINE                        CHAR(80)  EXT.                   ERR00070
        CNT                         FIXED(2)  EXT.                   ERR00080
        CRD_CTR                     FIXED(3)  EXT.                   ERR00090
        CHK                         CHAR(1).                         ERR00100
        ERR1                        FIXED(1).                        ERR00110
        LABEL                       FIXED(2).                        ERR00120
        TAIL                        CHAR(8):                         ERR00130
                                                                      ERR00140
        CHK = SUBSTR (LINE, 1, 1):                                    ERR00150
        ERR1 = VERIFY (CHK, '0123456789'):                            ERR00160
        IF ERR1 ¬= 0 THEN GO TO EC001:                                ERR00170
                                                                      ERR00180
    EC002: LABEL = VERIFY (LINE, '0123456789'):                       ERR00190
        IF LABEL > 6 THEN GO TO EC003:                                ERR00200
    EC004: TAIL = SUBSTR (LINE, 73, 8):                               ERR00210
        IF TAIL ¬= ' ' THEN GO TO EC005:                              ERR00220
    EC006: RETURN:                                                    ERR00230
                                                                      ERR00240
    EC001: ERROR(CNT) = 'NUMERIC NOT FOUND IN COLUMN ONE':            ERR00250
        SRS_LIN(CNT) = CRD_CTR:                                       ERR00260
        CNT = CNT + 1:                                                ERR00270
        GO TO EC002:                                                  ERR00280
                                                                      ERR00290
    EC003: ERROR(CNT) = 'LINE NUMBER EXCEEDS FIVE DIGITS':            ERR00300
        SRS_LIN(CNT) = CRD_CTR:                                       ERR00310
        CNT = CNT + 1:                                                ERR00320
        GO TO EC004:                                                  ERR00330
    EC005: ERROR(CNT) = 'CHARACTERS FOUND BEYOND COLUMN 72':          ERR00340
        SRS_LIN(CNT) = CRD_CTR:                                       ERR00350
        CNT = CNT + 1:                                                ERR00360
        GO TO EC006:                                                  ERR00370
END ERRCHK:                                                           ERR00380
```

FILE: REMARK PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
REMARK : PROCEDURE (K):                                                 REM00010
    DCL  1    PRS_TBL                    (999)     EXT,                 REM00020
               2    TOKEN                CHAR      (10),                REM00030
               2    CARD                 FIXED     (4):                 REM00040
                                                                        REM00050
    DCL  K                               FIXED     (4):                 REM00060
                                                                        REM00070
    RM001:K = K + 1;                                                    REM00080
          IF TOKEN(K) = '$      ' THEN RETURN;                          REM00090
          GO TO RM001;                                                  REM00100
END REMARK;                                                             REM00110
```

FILE: KEYREC PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
KEYREC : PROCEDURE (K):                                                 KEY00010
                                                                        KEY00020
    DCL  1   PRS_TBL                (999)    EXT,                      KEY00030
             2   TOKEN              CHAR     (10),                     KEY00040
             2   CARD               FIXED    (4);                      KEY00050
    DCL  1   KEY_TBL                (30)     EXT,                      KEY00060
             2   KY_NTRY            CHAR     (10);                     KEY00070
    DCL  1   UST                    (999)    EXT,                      KEY00080
             2   US_NTRY            CHAR     (3),                      KEY00090
             2   UST_PTR            FIXED    (4);                      KEY00100
                                                                        KEY00110
    DCL  N                          FIXED    (4),                      KEY00120
         K                          FIXED    (4),                      KEY00130
         UST_IDX                    FIXED    (4)  EXT;                 KEY00140
                                                                        KEY00150
         N = 1;                                                         KEY00160
         DO WHILE (KY_NTRY(N) ¬= ' ');                                  KEY00170
              IF TOKEN(K) = KY_NTRY(N) THEN GO TO KR001;                KEY00180
              N = N + 1;                                                KEY00190
         END;                                                           KEY00200
         RETURN;                                                        KEY00210
                                                                        KEY00220
    KR001: US_NTRY(UST_IDX) = 'KEY';                                    KEY00230
         UST_PTR(UST_IDX) = N;                                          KEY00240
         UST_IDX = UST_IDX + 1;                                         KEY00250
         RETURN;                                                        KEY00260
END KEYREC;                                                             KEY00270
```

FILE: TRMREC PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
TRMREC : PROCEDURE (K):                                                 TRM00010
                                                                        TRM00020
   DCL  1   PRS_TBL                  (999)     EXT,                    TRM00030
            2   TOKEN                CHAR      (10),                   TRM00040
            2   CARD                 FIXED     (4);                    TRM00050
   DCL  1   TRM_TBL                  (30)      EXT,                    TRM00060
            2   TM_NTRY              CHAR      (3),                    TRM00070
            2   TRM_PRI              FIXED     (1);                    TRM00080
   DCL  1   UST                      (999)     EXT,                    TRM00090
            2   US_NTRY              CHAR      (3),                    TRM00100
            2   UST_PTR              FIXED     (4);                    TRM00110
                                                                        TRM00120
   DCL  N                            FIXED     (4),                    TRM00130
        K                            FIXED     (4),                    TRM00140
        UST_IDX                      FIXED     (4)  EXT;               TRM00150
                                                                        TRM00160
        N = 1;                                                          TRM00170
        DO WHILE (TM_NTRY(N) ^= ' ');                                   TRM00180
           IF TOKEN(K) = TM_NTRY(N) THEN GO TO TR001;                   TRM00190
           N = N + 1;                                                   TRM00200
        END;                                                            TRM00210
        RETURN;                                                         TRM00220
TR001:  US_NTRY(UST_IDX) = 'TRM';                                       TRM00230
        UST_PTR(UST_IDX) = N;                                           TRM00240
        UST_IDX = UST_IDX + 1;                                          TRM00250
        RETURN;                                                         TRM00260
END TRMREC;                                                             TRM00270
```

FILE: IDNREC PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
IDNREC : PROCEDURE (K):                                                 IDN00010
                                                                        IDN00020
   DCL  1   PRS_TBL                    (999)     EXT,                  IDN00030
            2   TOKEN                  CHAR      (10),                 IDN00040
            2   CARD                   FIXED     (4);                  IDN00050
   DCL  1   IDN_TBL                    (99)      EXT,                  IDN00060
            2   ID_NTRY                CHAR      (2);                  IDN00070
   DCL  1   UST                        (999)     EXT,                  IDN00080
            2   US_NTRY                CHAR      (3),                  IDN00090
            2   UST_PTR                FIXED     (4);                  IDN00100
                                                                        IDN00110
   DCL  N                              FIXED     (4),                  IDN00120
        K                              FIXED     (4),                  IDN00130
        UST_IDX                        FIXED     (4)  EXT,             IDN00140
        A                              FIXED     (2),                  IDN00150
        B                              CHAR (10);                      IDN00160
                                                                        IDN00170
        N = 1;                                                          IDN00180
        DO WHILE (ID_NTRY(N) ¬= '  ');                                  IDN00190
             IF TOKEN(K) = ID_NTRY(N) THEN GO TO IR001;                 IDN00200
             N = N + 1;                                                 IDN00210
        END;                                                            IDN00220
                                                                        IDN00230
   /* DOES TOKEN SATISFY DEFINITION OF IDENTIFIER ? */                  IDN00240
                                                                        IDN00250
        A = VERIFY(TOKEN(K), 'ABCDEFGHIJKLMNOPQRSTUVWXYZ');             IDN00260
        IF A = 1 THEN RETURN;                                           IDN00270
        B = SUBSTR(TOKEN(K), 2);                                        IDN00280
        A = VERIFY(B, '0123456789 ');                                   IDN00290
        IF A ¬= 0 THEN RETURN;                                          IDN00300
        A = INDEX(TOKEN(K), ' ');                                       IDN00310
        IF A > 3 THEN RETURN;                                           IDN00320
        IF A = 0 THEN RETURN;                                           IDN00330
                                                                        IDN00340
   /* MAKE NEW ENTRY IN IDENTIFIER TABLE */                            IDN00350
                                                                        IDN00360
        ID_NTRY(N) = TOKEN(K);                                          IDN00370
                                                                        IDN00380
   /* WRITE INTO UNIFORM SYMBOL TABLE */                               IDN00390
                                                                        IDN00400
   IR001:US_NTRY(UST_IDX) = 'IDN';                                      IDN00410
        UST_PTR(UST_IDX) = N;                                           IDN00420
        UST_IDX = UST_IDX + 1;                                          IDN00430
        RETURN;                                                         IDN00440
END IDNREC;                                                             IDN00450
```

FILE: LITREC PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
LITREC : PROCEDURE (K):                                                 LIT00010
                                                                        LIT00020
   DCL  1   PRS_TBL                (999)      EXT,                     LIT00030
            2   TOKEN               CHAR       (10),                    LIT00040
            2   CARD                FIXED      (4);                     LIT00050
   DCL  1   LIT_TBL                 (500)       EXT,                    LIT00060
            2   LT_NTRY             CHAR       (9);                     LIT00070
   DCL 1    UST                     (999)      EXT,                     LIT00080
            2   US_NTRY             CHAR       (3),                     LIT00090
            2   UST_PTR             FIXED      (4);                     LIT00100
                                                                        LIT00110
   DCL  N                           FIXED      (4),                     LIT00120
        K                           FIXED      (4),                     LIT00130
        UST_IDX                     FIXED      (4)  EXT,                LIT00140
        A                           FIXED      (2),                     LIT00150
        B                           CHAR       (20);                    LIT00160
                                                                        LIT00170
        N = 1;                                                          LIT00180
        DO WHILE (LT_NTRY(N) ¬= ' ');                                   LIT00190
            IF TOKEN(K) = LT_NTRY(N) THEN GO TO LR001;                  LIT00200
            N = N + 1;                                                  LIT00210
        END;                                                            LIT00220
                                                                        LIT00230
   /* DOES TOKEN SATISFY DEFINITION OF LITERAL ? */                     LIT00240
                                                                        LIT00250
        A = VERIFY(TOKEN(K), '0123456789 .');                           LIT00260
        IF A ¬= 0 THEN RETURN;                                          LIT00270
        A = INDEX(TOKEN(K), '.');                                       LIT00280
        IF A = 0 THEN GO TO LR002;                                      LIT00290
        B = SUBSTR(TOKEN(K), A+1);                                      LIT00300
        A = INDEX(B, '.');                                              LIT00310
        IF A ¬= 0 THEN RETURN;                                          LIT00320
   LR002:A = INDEX(TOKEN(K), ' ');                                      LIT00330
        IF A > 10 THEN RETURN;                                          LIT00340
        IF A = 0 THEN RETURN;                                           LIT00350
                                                                        LIT00360
   /* MAKE A NEW ENTRY IN LITERAL TABLE */                              LIT00370
                                                                        LIT00380
        LT_NTRY(N) = TOKEN(K);                                          LIT00390
                                                                        LIT00400
   /* WRITE INTO UNIFORM SYMBOL TABLE */                                LIT00410
                                                                        LIT00420
   LR001:US_NTRY(UST_IDX) = 'LIT';                                      LIT00430
        UST_PTR(UST_IDX) = N;                                           LIT00440
        UST_IDX = UST_IDX + 1;                                          LIT00450
        RETURN;                                                         LIT00460
END LITREC;                                                             LIT00470
```

FILE: HEXREC PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
HEXREC : PROCEDURE (K);                                                 HEX00010
                                                                        HEX00020
   DCL  1   PRS_TBL                  (999)    EXT,                     HEX00030
            2   TOKEN                CHAR     (10),                    HEX00040
            2   CARD                 FIXED    (4);                     HEX00050
   DCL  1   LIT_TBL                  (500)     EXT,                    HEX00060
            2   LT_NTRY              CHAR     (9);                     HEX00070
   DCL  1   UST                      (999)    EXT,                     HEX00080
            2   US_NTRY              CHAR     (3),                     HEX00090
            2   UST_PTR              FIXED    (4);                     HEX00100
                                                                        HEX00110
   DCL  K                            FIXED    (4),                     HEX00120
        UST_IDX                      FIXED    (4)  EXT,                HEX00130
        N                            FIXED    (4),                     HEX00140
        A                            FIXED    (2);                     HEX00150
                                                                        HEX00160
        N = 1;                                                          HEX00170
        DO WHILE (LT_NTRY(N) ¬= ' ');                                   HEX00180
           IF TOKEN(K) = LT_NTRY(N) THEN GO TO HR001;                   HEX00190
           N = N + 1;                                                   HEX00200
        END;                                                            HEX00210
                                                                        HEX00220
   /* DOES TOKEN SATISFY DEFINITION OF HEXADECIMAL LITERAL ? */         HEX00230
                                                                        HEX00240
        A = VERIFY(TOKEN(K), '0123456789ABCDEF ');                      HEX00250
        IF A ¬= 0 THEN RETURN;                                          HEX00260
        A = INDEX(TOKEN(K), '.');                                       HEX00270
        IF A ¬= 0 THEN RETURN;                                          HEX00280
        A = INDEX(TOKEN(K), ' ');                                       HEX00290
        IF A ¬= 5 THEN RETURN;                                          HEX00300
                                                                        HEX00310
   /* MAKE A NEW ENTRY IN LITERAL TABLE */                              HEX00320
                                                                        HEX00330
        LT_NTRY(N) = TOKEN(K);                                          HEX00340
                                                                        HEX00350
   /* WRITE INTO UNIFORM SYMBOL TABLE */                                HEX00360
                                                                        HEX00370
   HR001:US_NTRY(UST_IDX) = 'LIT';                                      HEX00380
        UST_PTR(UST_IDX) = N;                                           HEX00390
        UST_IDX = UST_IDX + 1;                                          HEX00400
        RETURN;                                                         HEX00410
END HEXREC;                                                             HEX00420
```

FILE: TGTCHK PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
TGTCHK : PROCEDURE:                                                    TGT00010
    DCL  1   UST                          (999)    EXT,                TGT00020
             2   US_NTRY                  CHAR     (3),                TGT00030
             2   UST_PTR                  FIXED    (4);                TGT00040
    DCL  1   ERR_TBL                      (75)     EXT,                TGT00050
             2   ERROR                    CHAR     (40),               TGT00060
             2   SRS_LIN                  FIXED    (3);                TGT00070
    DCL  1   LIT_TBL                      (500)    EXT,                TGT00080
             2   LT_NTRY                  CHAR     (9);                TGT00090
    DCL  UST_IDX                          FIXED    (4)  EXT,           TGT00100
         CNT                              FIXED    (2)  EXT,           TGT00110
         CRD_CTR                          FIXED    (3)  EXT,           TGT00120
         TGT_PTR                          FIXED    (4)  EXT;           TGT00130
                                                                       TGT00140
    /* IS THE FIRST UNIFORM SYMBOL A VALID LINE NUMBER ? */            TGT00150
                                                                       TGT00160
        IF US_NTRY(UST_IDX) ¬= 'LIT' THEN GO TO TC001;                 TGT00170
                                                                       TGT00180
    /* CHECK FOR ALL NUMERICS */                                       TGT00190
                                                                       TGT00200
        IF(VERIFY(LT_NTRY(UST_PTR(UST_IDX)),'0123456789 ')¬=0)         TGT00210
            THEN GO TO TC001;                                          TGT00220
                                                                       TGT00230
    /* LINE NUMBER OK */                                               TGT00240
        TGT_PTR = UST_PTR(UST_IDX);                                    TGT00250
        UST_IDX = UST_IDX + 1;                                         TGT00260
        RETURN;                                                        TGT00270
    TC001: ERROR(CNT) = 'LINE DOES NOT START WITH A VALID NUMBER';     TGT00280
        SRS_LIN(CNT) = CRD_CTR;                                        TGT00290
        CNT = CNT + 1;                                                 TGT00300
        DO WHILE ((US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23));   TGT00310
            UST_IDX = UST_IDX + 1;                                     TGT00320
        END;                                                           TGT00330
        UST_IDX = UST_IDX + 1;                                         TGT00340
        CRD_CTR = CRD_CTR + 1;                                         TGT00350
        RETURN;                                                        TGT00360
END TGTCHK;                                                            TGT00370
```

```
FILE: LETAR     PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

 LETAR : PROCEDURE:                                                          LET00010
    DCL  1   UST                          (999)     EXT,                   LET00020
             2   US_NTRY                  CHAR      (3),                   LET00030
             2   UST_PTR                  FIXED     (4);                   LET00040
    DCL  1   ERR_TBL                      (75)      EXT,                   LET00050
             2   ERROR                    CHAR      (40),                  LET00060
             2   SRS_LIN                  FIXED     (3);                   LET00070
    DCL  1   TRM_TBL                      (30)      EXT,                   LET00080
             2   TM_NTRY                  CHAR      (3),                   LET00090
             2   TRM_PRI                  FIXED     (1);                   LET00100
    DCL  1   MTX_FIL,                                                       LET00110
             2   LBL_PTR                  CHAR      (4),                   LET00120
             2   OPR                      CHAR      (3),                   LET00130
             2   OPR_PTR                  CHAR      (4),                   LET00140
             2   OP1                      CHAR      (3),                   LET00150
             2   OP1_PTR                  CHAR      (4),                   LET00160
             2   OP2                      CHAR      (3),                   LET00170
             2   OP2_PTR                  CHAR      (4),                   LET00180
             2   FILL                     CHAR      (55) INIT((55)' '):    LET00190
    DCL  1   OPND                         CONTROLLED,                      LET00200
             2   OPND_STACK_TBL           CHAR      (3),                   LET00210
             2   OPND_STACK_PTR           FIXED     (4);                   LET00220
    DCL  1   OPTR                         CONTROLLED,                      LET00230
             2   OPTR_STACK_TBL           CHAR      (3),                   LET00240
             2   OPTR_STACK_PTR           FIXED     (4),                   LET00250
             2   OPTR_PRIORITY            FIXED     (4);                   LET00260
    DCL  1   FCN_OPND                     CONTROLLED,                      LET00270
             2   FCN_OPND_STACK_TBL       CHAR      (3),                   LET00280
             2   FCN_OPND_STACK_PTR       FIXED     (4);                   LET00290
    DCL  1   FCN_OPTR                     CONTROLLED,                      LET00300
             2   FCN_OPTR_STACK_TBL       CHAR      (3),                   LET00310
             2   FCN_OPTR_STACK_PTR       FIXED     (4),                   LET00320
             2   FCN_OPTR_PRIORITY        FIXED     (4);                   LET00330
    DCL  UST_IDX                          FIXED     (4)  EXT,              LET00340
         CNT                              FIXED     (2)  EXT,              LET00350
         AREA                             CHAR      (7),                   LET00360
         ZERO0                            CHAR      (1)  INIT('0'),        LET00370
         BLANK                            CHAR      (1)  INIT(' '),        LET00380
         CRD_CTR                          FIXED     (3)  EXT,              LET00390
         TGT_PTR                          FIXED     (4)  EXT,              LET00400
         TMP_STR                          FIXED     (3)  EXT,              LET00410
         NO_FLG                           FIXED     (1),                   LET00420
         PAR_CT                           FIXED     (3),                   LET00430
         PRIORITY                         FIXED     (3),                   LET00440
         FCN_PAR_CT                       FIXED     (3),                   LET00450
         FCN_FLG                          FIXED     (1),                   LET00460
         ASCN                             FIXED     (4),                   LET00470
         FCN_PTR                          FIXED     (4);                   LET00480
    DCL  MATRIX                           FILE      RECORD;                LET00490
                                                                            LET00500
    NO_FLG, PAR_CT, PRIORITY, FCN_PAR_CT, FCN_FLG, TMP_STR = 0;             LET00510
                                                                            LET00520
    /* NEXT SYMBOL MUST BE AN IDENTIFIER */                                 LET00530
                                                                            LET00540
    UST_IDX = UST_IDX + 1;                                                  LET00550
```

```
FILE: LETAR    PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

        IF US_NTRY(UST_IDX)¬='IDN' THEN                                    LET00560
            DO: ERROR(CNT)= 'IDENTIFIER MUST FOLLOW LET':                  LET00570
                GO TO LA001_ERROR:                                         LET00580
            END:                                                           LET00590
        ASGN = UST_PTR(UST_IDX):                                           LET00600
                                                                           LET00610
    /* NEXT SYMBOL MUST BE '=' */                                          LET00620
                                                                           LET00630
        UST_IDX = UST_IDX + 1:                                             LET00640
        IF(US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=6) THEN             LET00650
            DO: ERROR(CNT)='EQUAL SIGN IS NOT IN PROPER POSITION':         LET00660
                GO TO LA001_ERROR:                                         LET00670
            END:                                                           LET00680
                                                                           LET00690
    /* CHECK FOR LEFT PARENTHESIS, RIGHT PARENTHESIS, LINE END */          LET00700
                                                                           LET00710
LA002:  UST_IDX = UST_IDX + 1:                                             LET00720
        IF(US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=7) THEN               LET00730
            DO:                                                            LET00740
                IF FCN_FLG = 1 THEN FCN_PAR_CT = FCN_PAR_CT + 10:          LET00750
                PAR_CT = PAR_CT + 10:                                      LET00760
                GO TO LA002:                                               LET00770
            END:                                                           LET00780
LA003:  IF(US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=8) THEN               LET00790
            DO:                                                            LET00800
                IF FCN_FLG=1 THEN FCN_PAR_CT = FCN_PAR_CT-10:              LET00810
                PAR_CT = PAR_CT - 10:                                      LET00820
                UST_IDX = UST_IDX + 1:                                     LET00830
                GO TO LA003:                                               LET00840
            END:                                                           LET00850
        IF(US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=23) THEN              LET00860
            DO:                                                            LET00870
                NO_FLG = 1:                                                LET00880
                GO TO LA004_POP:                                           LET00890
            END:                                                           LET00900
                                                                           LET00910
    /* IS SYMBOL A FUNCTION NAME ? */                                      LET00920
                                                                           LET00930
        IF(US_NTRY(UST_IDX) ='TRM')&(UST_PTR(UST_IDX)>12) &                LET00940
            (UST_PTR(UST_IDX)<23) THEN                                     LET00950
            DO:                                                            LET00960
                FCN_FLG = 1:                                               LET00970
                FCN_PTR = UST_PTR(UST_IDX):                                LET00980
                UST_IDX = UST_IDX + 1:                                     LET00990
                IF(US_NTRY(UST_IDX)¬='TRM')|                               LET01000
                    (UST_PTR(UST_IDX)¬=7) THEN                             LET01010
                    DO:                                                    LET01020
                ERROR(CNT)='PARENTHESIS MUST ENCLOSE FUNCTION ARG':        LET01030
                        GO TO LA001_ERROR:                                 LET01040
                    END:                                                   LET01050
                UST_IDX = UST_IDX + 1:                                     LET01060
                GO TO LA002:                                               LET01070
            END:                                                           LET01080
                                                                           LET01090
    /* IS SYMBOL A LITERAL OR IDENTIFIER ? */                              LET01100
```

```
FILE: LETAR     PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

                                                                                LET01110
        IF(US_NTRY(UST_IDX)¬='LIT')&(US_NTRY(UST_IDX)¬='IDN') THEN              LET01120
            DO:                                                                 LET01130
                ERROR(CNT)='INVALID SYNTAX';                                    LET01140
                GO TO LA001_ERROR;                                              LET01150
            END;                                                                LET01160
                                                                                LET01170
    /* PUSH SYMBOL ONTO APPROPRIATE STACK */                                    LET01180
                                                                                LET01190
        IF FCN_FLG = 1 THEN                                                     LET01200
            DO:                                                                 LET01210
                ALLOCATE FCN_OPND;                                              LET01220
                FCN_OPND_STACK_TAL = US_NTRY(UST_IDX);                          LET01230
                FCN_OPND_STACK_PTR = UST_PTR(UST_IDX);                          LET01240
                GO TO LA005;                                                    LET01250
            END;                                                                LET01260
        ALLOCATE OPND;                                                          LET01270
        OPND_STACK_TAL = US_NTRY(UST_IDX);                                      LET01280
        OPND_STACK_PTR = UST_PTR(UST_IDX);                                      LET01290
                                                                                LET01300
    /* CHECK FOR LEFT PARENTHESIS, RIGHT PARENTHESIS, LINE END */               LET01310
                                                                                LET01320
LA005:  UST_IDX = UST_IDX + 1;                                                  LET01330
        IF(US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=7) THEN                    LET01340
            DO:                                                                 LET01350
                IF FCN_FLG = 1 THEN FCN_PAR_CT = FCN_PAR_CT+10;                 LET01360
                PAR_CT = PAR_CT + 10;                                           LET01370
                GO TO LA005;                                                    LET01380
            END;                                                                LET01390
LA006:  IF(US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=8) THEN                    LET01400
            DO:                                                                 LET01410
                IF FCN_FLG = 1 THEN FCN_PAR_CT = FCN_PAR_CT-10;                 LET01420
                PAR_CT = PAR_CT -10;                                            LET01430
                IF(FCN_PAR_CT=0)&(FCN_FLG=1) THEN                               LET01440
                  DO:                                                           LET01450
                    IF(ALLOCATION(FCN_OPTR)=0) THEN                             LET01460
                      GO TO LA007_POP_FCN;                                      LET01470
                    GO TO LA004_POP;                                            LET01480
                  END;                                                          LET01490
                UST_IDX = UST_IDX + 1;                                          LET01500
                GO TO LA006;                                                    LET01510
            END;                                                                LET01520
        IF(US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=23) THEN                   LET01530
            DO:                                                                 LET01540
                NO_FLG = 1;                                                     LET01550
                IF ALLOCATION(OPTR)=0 THEN GO TO LA009_FINISH;                  LET01560
                GO TO LA004_POP;                                                LET01570
            END;                                                                LET01580
                                                                                LET01590
    /* IS SYMBOL AN OPERATOR ? */                                               LET01600
        IF(US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)>5) THEN                   LET01610
            DO:                                                                 LET01620
                ERROR(CNT)='INVALID SYNTAX';                                    LET01630
                GO TO LA001_ERROR;                                              LET01640
            END;                                                                LET01650
```

```
          PRIORITY=TRM_PRI(UST_PTR(UST_IDX))+PAR_CT:                    LET01660
                                                                        LET01670
     /* IS OPERATOR STACK EMPTY ? */                                    LET01680
          IF FCN_FLG = 1 THEN                                           LET01690
               DO:                                                      LET01700
                    IF ALLOCATION(FCN_OPTR)=0 THEN                      LET01710
                         GO TO LA008_PUSH_OPTR:                         LET01720
                    GO TO LA010:                                        LET01730
               END:                                                     LET01740
          IF ALLOCATION(OPTR)=0 THEN GO TO LA008_PUSH_OPTR:             LET01750
                                                                        LET01760
     /* IS TOP OF STACK GREATER THAN PRIORITY ? */                      LET01770
                                                                        LET01780
LA010:    IF FCN_FLG = 1 THEN                                           LET01790
               DO:                                                      LET01800
                    IF FCN_OPTR_PRIORITY > PRIORITY THEN                LET01810
                         GO TO LA004_POP:                               LET01820
                    GO TO LA008_PUSH_OPTR:                              LET01830
               END:                                                     LET01840
          IF OPTR_PRIORITY > PRIORITY THEN                              LET01850
               GO TO LA004_POP:                                         LET01860
          GO TO LA008_PUSH_OPTR:                                        LET01870
                                                                        LET01880
     /* POP STACKS AND WRITE INTO MATRIX */                             LET01890
                                                                        LET01900
LA004_POP:                                                              LET01910
          IF FCN_FLG = 1 THEN                                           LET01920
               DO:                                                      LET01930
                    AREA=TGT_PTR:                                       LET01940
                    AREA=TRANSLATE(AREA, ZERO0, BLANK):                 LET01950
                    LBL_PTR=SUBSTR(AREA, 4, 4):                         LET01960
                    OPR=FCN_OPTR_STACK_TBL:                             LET01970
                    AREA = FCN_OPTR_STACK_PTR:                          LET01980
                    AREA=TRANSLATE(AREA, ZERO0, BLANK):                 LET01990
                    OPR_PTR = SUBSTR(AREA, 4, 4):                       LET02000
                    FREE FCN_OPTR:                                      LET02010
                    OP2=FCN_OPND_STACK_TBL:                             LET02020
                    AREA=FCN_OPND_STACK_PTR:                            LET02030
                    AREA=TRANSLATE(AREA, ZERO0, BLANK):                 LET02040
                    OP2_PTR = SUBSTR(AREA, 4, 4):                       LET02050
                    FREE FCN_OPND:                                      LET02060
                    OP1=FCN_OPND_STACK_TBL:                             LET02070
                    AREA=FCN_OPND_STACK_PTR:                            LET02080
                    AREA=TRANSLATE(AREA, ZERO0, BLANK):                 LET02090
                    OP1_PTR= SUBSTR(AREA, 4, 4):                        LET02100
                    FREE FCN_OPND:                                      LET02110
                    WRITE FILE (MATRIX) FROM (MTX_FIL):                 LET02120
                    GO TO LA011:                                        LET02130
               END:                                                     LET02140
          AREA=TGT_PTR:                                                 LET02150
          AREA=TRANSLATE(AREA, ZERO0, BLANK):                           LET02160
          LBL_PTR=SUBSTR(AREA, 4, 4):                                   LET02170
          OPR=OPTR_STACK_TBL:                                           LET02180
          AREA=OPTR_STACK_PTR:                                          LET02190
          AREA=TRANSLATE(AREA, ZERO0, BLANK):                           LET02200
```

```
        OPR_PTR=SUBSTR(AREA, 4, 4);                                    LET02210
        FREE OPTR;                                                     LET02220
        OP2=OPND_STACK_TBL;                                            LET02230
        AREA=OPND_STACK_PTR;                                           LET02240
        AREA=TRANSLATE(AREA, ZEROO, BLANK);                            LET02250
        OP2_PTR=SUBSTR(AREA, 4, 4);                                    LET02260
        FREE OPND;                                                     LET02270
        OP1=OPND_STACK_TBL;                                            LET02280
        AREA=OPND_STACK_PTR;                                           LET02290
        AREA=TRANSLATE(AREA, ZEROO, BLANK);                            LET02300
        OP1_PTR=SUBSTR(AREA, 4, 4);                                    LET02310
        FREE OPND;                                                     LET02320
        WRITE FILE (MATRIX) FROM (MTX_FIL);                            LET02330
                                                                       LET02340
    /* ASSIGN TEMPORARY VALUE AND PUSH ONTO APPROPRIATE STACK*/        LET02350
                                                                       LET02360
LA011:TMP_STR=TMP_STR + 1;                                             LET02370
        IF FCN_FLG = 1 THEN                                            LET02380
            DO:                                                        LET02390
                ALLOCATE FCN_OPND;                                     LET02400
                FCN_OPND_STACK_TBL = 'TMP';                            LET02410
                FCN_OPND_STACK_PTR=TMP_STR;                            LET02420
                GO TO LA012;                                           LET02430
            END;                                                       LET02440
        ALLOCATE OPND;                                                 LET02450
        OPND_STACK_TBL = 'TMP';                                        LET02460
        OPND_STACK_PTR=TMP_STR;                                        LET02470
                                                                       LET02480
    /* CHECK FOR END OF FUNCTION */                                    LET02490
                                                                       LET02500
LA012:IF(FCN_PAR_CT=0)&(FCN_FLG=1) THEN                                LET02510
            DO:                                                        LET02520
                IF ALLOCATION(FCN_OPTR)=0 THEN                         LET02530
                    GO TO LA007_POP_FCN;                               LET02540
                GO TO LA004_POP;                                       LET02550
            END;                                                       LET02560
                                                                       LET02570
    /* IS OPERATOR STACK EMPTY AND END FLAG SET ? */                   LET02580
                                                                       LET02590
        IF (ALLOCATION(OPTR)=0)&(NO_FLG=1) THEN                        LET02600
            GO TO LA009_FINISH;                                        LET02610
                                                                       LET02620
    /* CHECK FOR END OF LINE */                                        LET02630
                                                                       LET02640
        IF(NO_FLG=1) THEN                                              LET02650
            GO TO LA004_POP;                                           LET02660
                                                                       LET02670
    /* CHECK FOR EMPTY OPERATOR STACK */                               LET02680
                                                                       LET02690
        IF FCN_FLG = 1 THEN                                            LET02700
            DO:                                                        LET02710
                IF ALLOCATION(FCN_OPTR)=0 THEN                         LET02720
                    GO TO LA008_PUSH_OPTR;                             LET02730
                GO TO LA010;                                           LET02740
            END;                                                       LET02750
```

```
FILE: LETAR     PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

        IF ALLOCATION(OPTR)=0 THEN                                      LET02760
            GO TO LA008_PUSH_OPTR:                                      LET02770
        GO TO LA010:                                                    LET02780
                                                                        LET02790
    /* PUSH OPERATOR ONTO APPROPRIATE STACK */                          LET02800
                                                                        LET02810
LA008_PUSH_OPTR:                                                        LET02820
        IF FCN_FLG = 1 THEN                                             LET02830
            DO:                                                         LET02840
                ALLOCATE FCN_OPTR:                                      LET02850
                FCN_OPTR_STACK_TBL = US_NTRY(UST_IDX):                  LET02860
                FCN_OPTR_STACK_PTR = UST_PTR(UST_IDX):                  LET02870
                FCN_OPTR_PRIORITY = PRIORITY:                           LET02880
                GO TO LA002:                                            LET02890
            END:                                                        LET02900
        ALLOCATE OPTR:                                                  LET02910
        OPTR_STACK_TBL = US_NTRY(UST_IDX):                              LET02920
        OPTR_STACK_PTR = UST_PTR(UST_IDX):                              LET02930
        OPTR_PRIORITY = PRIORITY:                                       LET02940
        GO TO LA002:                                                    LET02950
                                                                        LET02960
    /* FINISH FUNCTION PROCESSING */                                    LET02970
                                                                        LET02980
LA007_PRP_FCN:                                                          LET02990
        AREA=TGT_PTR:                                                   LET03000
        AREA=TRANSLATE(AREA, ZERO0, BLANK):                             LET03010
        TBL_PTR = SUBSTR (AREA, 4, 4):                                  LET03020
        OPR='TRM':                                                      LET03030
        AREA = FCN_PTR:                                                 LET03040
        AREA=TRANSLATE(AREA, ZERO0, BLANK):                             LET03050
        OP2_PTR=SUBSTR(AREA, 4, 4):                                     LET03060
        OP1=FCN_OPND_STACK_TBL:                                         LET03070
        AREA=FCN_OPND_STACK_PTR:                                        LET03080
        AREA=TRANSLATE(AREA, ZERO0, BLANK):                             LET03090
        OP1_PTR=SUBSTR(AREA, 4, 4):                                     LET03100
        FREE FCN_OPND:                                                  LET03110
        OP2='    ':                                                     LET03120
        OP2_PTR='0000':                                                 LET03130
        WRITE FILE(MATRIX) FROM (MTX_FIL):                              LET03140
        FCN_FLG=0:                                                      LET03150
        TMP_STR=TMP_STR + 1:                                            LET03160
        ALLOCATE OPND:                                                  LET03170
        OPND_STACK_TBL = 'TMP':                                         LET03180
        OPND_STACK_PTR = TMP_STR:                                       LET03190
        GO TO LA005:                                                    LET03200
                                                                        LET03210
    /* FINISH ASSIGNMENT STATEMENT AND RETURN TO CONTROL */             LET03220
                                                                        LET03230
LA009_FINISH:                                                           LET03240
                                                                        LET03250
    /* CHECK FOR AGREEMENT OF PARENTHESES */                            LET03260
                                                                        LET03270
        IF PAR_CT¬=0 THEN                                               LET03280
            DO:                                                         LET03290
              ERROR(CNT)='UNEQUAL NO. OF LEFT AND RIGHT PAREN.':        LET03300
```

```
                GO TO LA001_ERROR;                                      LET03310
                END;                                                    LET03320
          AREA=TGT_PTR;                                                 LET03330
          AREA=TRANSLATE(AREA, ZEROO, BLANK);                           LET03340
          LAL_PTR=SUBSTR(AREA, 4, 4);                                   LET03350
          OPR='TRM';                                                    LET03360
          OPR_PTR='0006';                                               LET03370
          OP2=OPND_STACK_TAL;                                           LET03380
          AREA=OPND_STACK_PTR;                                          LET03390
          AREA=TRANSLATE(AREA, ZEROO, BLANK);                           LET03400
          OP2_PTR=SUBSTR(AREA, 4, 4);                                   LET03410
          FREE OPND;                                                    LET03420
          OP1='IDN';                                                    LET03430
          AREA=ASGN;                                                    LET03440
          AREA=TRANSLATE(AREA, ZEROO, BLANK);                           LET03450
          OP1_PTR=SUBSTR(AREA, 4, 4);                                   LET03460
          WRITE FILE (MATRIX) FROM (MTX_FIL);                           LET03470
          UST_IDX = UST_IDX + 1;                                        LET03480
          CRD_CTR = CRD_CTR + 1;                                        LET03490
          RETURN;                                                       LET03500
                                                                        LET03510
     /* ERROR ROUTINE */                                               LET03520
                                                                        LET03530
LA001_ERROR:                                                            LET03540
          SRS_LIN(CNT)=CRD_CTR;                                         LET03550
          CNT=CNT+1;                                                    LET03560
          DO WHILE ((US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23));  LET03570
               UST_IDX = UST_IDX + 1;                                   LET03580
          END;                                                          LET03590
          UST_IDX = UST_IDX + 1;                                        LET03600
          CRD_CTR = CRD_CTR + 1;                                        LET03610
                                                                        LET03620
     /* CLEAR STACKS */                                                LET03630
                                                                        LET03640
          DO WHILE(ALLOCATION(OPND)¬=0);                                LET03650
               FREE OPND;                                               LET03660
          END;                                                          LET03670
          DO WHILE(ALLOCATION(OPTR)¬=0);                                LET03680
               FREE OPTR;                                               LET03690
          END;                                                          LET03700
          DO WHILE(ALLOCATION(FCN_OPND)¬=0);                            LET03710
               FREE FCN_OPND;                                           LET03720
          END;                                                          LET03730
          DO WHILE(ALLOCATION(FCN_OPTR)¬=0);                            LET03740
               FREE FCN_OPTR;                                           LET03750
          END;                                                          LET03760
          RETURN;                                                       LET03770
                                                                        LET03780
END LETAR;                                                              LET03790
```

FILE: RDINAR PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
RDINAR : PROCEDURE;                                                    RDI00010
                                                                       RDI00020
   DCL   1  UST                          (999)     EXT,              RDI00030
            2   US_NTRY                  CHAR      (3),              RDI00040
            2   UST_PTR                  FIXED     (4);              RDI00050
   DCL   1  ERR_TBL                      (75)      EXT,              RDI00060
            2   ERROR                    CHAR      (40),             RDI00070
            2   SRS_LIN                  FIXED     (3);              RDI00080
   DCL   1  LIT_TBL                      (500)     EXT,              RDI00090
            2   LT_NTRY                  CHAR      (9);              RDI00100
   DCL   1  MTX_FIL,                                                  RDI00110
            2   LBL_PTR                  CHAR      (4),              RDI00120
            2   OPR                      CHAR      (3),              RDI00130
            2   OPR_PTR                  CHAR      (4),              RDI00140
            2   OP1                      CHAR      (3),              RDI00150
            2   OP1_PTR                  CHAR      (4),              RDI00160
            2   OP2                      CHAR      (3),              RDI00170
            2   OP2_PTR                  CHAR      (4),              RDI00180
            2   FILL                     CHAR      (55) INIT((55)' '); RDI00190
   DCL   UST_IDX                         FIXED     (4)  EXT,         RDI00200
         AREA                            CHAR      (7),              RDI00210
         ZERO0                           CHAR      (1)  INIT('0'),   RDI00220
         BLANK                           CHAR      (1)  INIT(' '),   RDI00230
         CNT                             FIXED     (2)  EXT,         RDI00240
         CRD_CTR                         FIXED     (3)  EXT,         RDI00250
         TGT_PTR                         FIXED     (4)  EXT;         RDI00260
   DCL   MATRIX                          FILE      RECORD;           RDI00270
                                                                       RDI00280
   /* IS NEXT UNIFORM SYMBOL AN IDENTIFIER ? */                        RDI00290
                                                                       RDI00300
         UST_IDX = UST_IDX + 1;                                        RDI00310
         IF US_NTRY(UST_IDX)¬='IDN' THEN                               RDI00320
            DO;                                                        RDI00330
               ERROR(CNT)='VARIABLE NAME MUST FOLLOW RDIN';            RDI00340
               SRS_LIN(CNT)=CRD_CTR;                                   RDI00350
               CNT=CNT+1;                                              RDI00360
               DO WHILE((US_NTRY(UST_IDX)¬='TRM')|                     RDI00370
                    (UST_PTR(UST_IDX)¬=23));                           RDI00380
                  UST_IDX = UST_IDX + 1;                               RDI00390
               END;                                                    RDI00400
               UST_IDX = UST_IDX + 1;                                  RDI00410
               CRD_CTR = CRD_CTR + 1;                                  RDI00420
               RETURN;                                                 RDI00430
            END;                                                       RDI00440
                                                                       RDI00450
   /* WRITE IDENTIFIER INTO MATRIX */                                  RDI00460
                                                                       RDI00470
         AREA = TGT_PTR;                                               RDI00480
         AREA = TRANSLATE (AREA, ZERO0, BLANK);                        RDI00490
         LBL_PTR = SUBSTR (AREA, 4, 4);                                RDI00500
         OPR = 'KEY';                                                  RDI00510
         OPR_PTR = '0019';                                             RDI00520
         OP1 = 'IDN';                                                  RDI00530
         AREA = UST_PTR(UST_IDX);                                      RDI00540
         AREA = TRANSLATE (AREA, ZERO0, BLANK);                        RDI00550
```

```
FILE: RDINAR   PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

        OP1_PTR = SUBSTR (AREA, 4, 4);                                  RDI00560
                                                                        RDI00570
    /* IS NEXT UNIFORM SYMBOL 'FROM' ? */                               RDI00580
                                                                        RDI00590
        UST_IDX = UST_IDX + 1;                                          RDI00600
        IF(US_NTRY(UST_IDX)¬='KEY')|(UST_PTR (UST_IDX)¬=15) THEN        RDI00610
            DO;                                                         RDI00620
                ERROR(CNT)='FROM MUST FOLLOW VARIABLE NAME';            RDI00630
                SRS_LIN(CNT) = CRD_CTR;                                 RDI00640
                CNT = CNT + 1;                                          RDI00650
                DO WHILE ((US_NTRY(UST_IDX)¬='TRM')|                    RDI00660
                      (UST_PTR(UST_IDX)¬=23));                          RDI00670
                    UST_IDX = UST_IDX + 1;                              RDI00680
                END;                                                    RDI00690
                UST_IDX = UST_IDX + 1;                                  RDI00700
                CRD_CTR = CRD_CTR + 1;                                  RDI00710
                RETURN;                                                 RDI00720
            END;                                                        RDI00730
                                                                        RDI00740
    /* IS NEXT UNIFORM SYMBOL A HEX ADDRESS ? */                        RDI00750
                                                                        RDI00760
        UST_IDX=UST_IDX+1;                                              RDI00770
        IF(US_NTRY(UST_IDX)¬='LIT')|                                    RDI00780
            (INDEX(LT_NTRY(UST_PTR(UST_IDX)),',')¬=0)|                  RDI00790
            (INDEX(LT_NTRY(UST_PTR(UST_IDX)),' ')¬=5) THEN              RDI00800
            DO;                                                         RDI00810
                ERROR(CNT)='HEX ADDRESS MUST FOLLOW FROM';              RDI00820
                SRS_LIN(CNT)=CRD_CTR;                                   RDI00830
                CNT=CNT+1;                                              RDI00840
                DO WHILE((US_NTRY(UST_IDX)¬='TRM')|                     RDI00850
                      (UST_PTR(UST_IDX)¬=23));                          RDI00860
                    UST_IDX=UST_IDX+1;                                  RDI00870
                END;                                                    RDI00880
                UST_IDX=UST_IDX+1;                                      RDI00890
              CRD_CTR=CRD_CTR+1;                                        RDI00900
                RETURN;                                                 RDI00910
            END;                                                        RDI00920
                                                                        RDI00930
    /* WRITE HEX ADDRESS INTO MATRIX */                                 RDI00940
                                                                        RDI00950
        OP2='LIT';                                                      RDI00960
        AREA = UST_PTR(UST_IDX);                                        RDI00970
        AREA = TRANSLATE (AREA, ZERO0, BLANK);                          RDI00980
        OP2_PTR = SUBSTR (AREA, 4, 4);                                  RDI00990
        WRITE FILE (MATRIX) FROM (MTX_FIL);                             RDI01000
                                                                        RDI01010
    /* IS REMAINDER OF SOURCE LINE BLANK ? */                           RDI01020
                                                                        RDI01030
        UST_IDX=UST_IDX+1;                                              RDI01040
        IF (US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23) THEN        RDI01050
            DO;                                                         RDI01060
                ERROR(CNT)='MULTIPLE ENTRIES FOR HEX ADDRESS';          RDI01070
                SRS_LIN(CNT)=CRD_CTR;                                   RDI01080
                CNT=CNT+1;                                              RDI01090
                DO WHILE ((US_NTRY(UST_IDX)¬='TRM')|                    RDI01100
```

```
                    (UST_PTR(UST_IDX)¬=23)):                                RDI01110
                    UST_IDX=UST_IDX+1;                                      RDI01120
                END;                                                        RDI01130
                UST_IDX=UST_IDX+1;                                          RDI01140
                CRD_CTR=CRD_CTR+1;                                          RDI01150
                RETURN;                                                     RDI01160
            END;                                                            RDI01170
                                                                            RDI01180
    /*NORMAL RETURN TO CONTROL*/                                            RDI01190
                                                                            RDI01200
        UST_IDX=UST_IDX+1;                                                  RDI01210
        CRD_CTR=CRD_CTR+1;                                                  RDI01220
        RETURN;                                                             RDI01230
END RDINAR;                                                                 RDI01240
```

```
FILE: IFAR      PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

 IFAR : PROCEDURE:                                                          IFA00010
   DCL  1    UST                          (999)     EXT.                   IFA00020
              2    US_NTRY                CHAR      (3),                   IFA00030
              2    UST_PTR                FIXED     (4);                   IFA00040
   DCL  1    ERR_TBL                      (75)      EXT.                   IFA00050
              2    ERROR                  CHAR      (40),                  IFA00060
              2    SRS_LIN                FIXED     (3);                   IFA00070
   DCL  1    TRM_TBL                      (30)      EXT.                   IFA00080
              2    TM_NTRY                CHAR      (3),                   IFA00090
              2    TRM_PRI                FIXED     (1);                   IFA00100
   DCL  1    LIT_TBL                      (500)     EXT.                   IFA00110
              2    LT_NTRY                CHAR      (9);                   IFA00120
   DCL  1    MTX_FIL.                                                      IFA00130
              2    LBL_PTR                CHAR      (4),                   IFA00140
              2    OPR                    CHAR      (3),                   IFA00150
              2    OPR_PTR                CHAR      (4),                   IFA00160
              2    OP1                    CHAR      (3),                   IFA00170
              2    OP1_PTR                CHAR      (4),                   IFA00180
              2    OP2                    CHAR      (3),                   IFA00190
              2    OP2_PTR                CHAR      (4),                   IFA00200
              2    FILL                   CHAR      (55) INIT((55)' '):    IFA00210
   DCL  UST_IDX                           FIXED     (4)  EXT.              IFA00220
        ORIGINAL_IDX                      FIXED     (4),                   IFA00230
        CNT                               FIXED     (2)  EXT.              IFA00240
        AREA                              CHAR      (7),                   IFA00250
        F_LIN                             FIXED     (6),                   IFA00260
        ZERO0                             CHAR      (1)  INIT('0'),        IFA00270
        BLANK                             CHAR      (1)  INIT(' '),        IFA00280
        CRD_CTR                           FIXED     (3)  EXT.              IFA00290
        TGT_PTR                           FIXED     (4)  EXT.              IFA00300
        TMP_STR                           FIXED     (3)  EXT.              IFA00310
        NXT_LIN                           CHAR      (9),                   IFA00320
        N                                 FIXED     (4),                   IFA00330
        CXL                               FIXED     (4),                   IFA00340
        CXR                               FIXED     (4),                   IFA00350
        RELOP                             FIXED     (4);                   IFA00360
   DCL  CDX                               ENTRY,                           IFA00370
        LFTAR                             ENTRY,                           IFA00380
        RETRNAR                           ENTRY,                           IFA00390
        READAR                            ENTRY,                           IFA00400
        RESTRAR                           ENTRY,                           IFA00410
        STOPAR                            ENTRY,                           IFA00420
        ENDAR                             ENTRY,                           IFA00430
        WTOUTAR                           ENTRY,                           IFA00440
        RDINAR                            ENTRY,                           IFA00450
        GOTOAR                            ENTRY,                           IFA00460
        GOSUBAR                           ENTRY;                           IFA00470
   DCL  MATRIX                            FILE      RECORD;                IFA00480
                                                                           IFA00490
        TMP_STR=0;                                                         IFA00500
                                                                           IFA00510
    /* PROCESS LEFT HAND SIDE OF CONDITION */                              IFA00520
                                                                           IFA00530
        CALL CDX;                                                          IFA00540
        CXL=TMP_STR;                                                       IFA00550
```

```
                                                                        IFA00560
     /* CHECK FOR RELATIONAL OPERATOR AFTER FIRST ARGUMENT */           IFA00570
                                                                        IFA00580
        IF UST_PTR(UST_IDX) = 6  THEN DO:                               IFA00590
             RELOP = 27:                                                IFA00600
             GO TO IF000:                                               IFA00610
             END:                                                       IFA00620
        IF UST_PTR(UST_IDX) = 9  THEN DO:                               IFA00630
             RELOP = 9:                                                 IFA00640
             GO TO IF000:                                               IFA00650
             END:                                                       IFA00660
        IF UST_PTR(UST_IDX) = 10 THEN DO:                               IFA00670
             RELOP = 10:                                                IFA00680
             GO TO IF000:                                               IFA00690
             END:                                                       IFA00700
        IF UST_PTR(UST_IDX) = 24 THEN DO:                               IFA00710
             RELOP = 24:                                                IFA00720
             GO TO IF000:                                               IFA00730
             END:                                                       IFA00740
        IF UST_PTR(UST_IDX)= 25 THEN DO:                                IFA00750
             RELOP = 25:                                                IFA00760
             GO TO IF000:                                               IFA00770
             END:                                                       IFA00780
        IF UST_PTR(UST_IDX) = 26 THEN DO:                               IFA00790
             RELOP = 26:                                                IFA00800
             GO TO IF000:                                               IFA00810
             END:                                                       IFA00820
        ERROR(CNT)='REL. OP. MUST FOLLOW FIRST CONDITION':              IFA00830
        GO TO IF003:                                                    IFA00840
                                                                        IFA00850
IF000:  /* PROCESS RIGHT HAND SIDE OF CONDITION */                      IFA00860
                                                                        IFA00870
        CALL CDX:                                                       IFA00880
        CXR=TMP_STR:                                                    IFA00890
                                                                        IFA00900
    /* WRITE MATRIX LINE */                                             IFA00910
                                                                        IFA00920
        AREA=TGT_PTR:                                                   IFA00930
        AREA = TRANSLATE(AREA, ZERO0, BLANK):                           IFA00940
        LBL_PTR=SUBSTR(AREA, 4, 4):                                     IFA00950
        OPR='TRM':                                                      IFA00960
        AREA=RELOP:                                                     IFA00970
        AREA = TRANSLATE(AREA, ZERO0, BLANK):                           IFA00980
        OPR_PTR = SUBSTR(AREA, 4 , 4):                                  IFA00990
        OP1 ='TMP':                                                     IFA01000
        AREA=CXL:                                                       IFA01010
        AREA = TRANSLATE(AREA, ZERO0, BLANK):                           IFA01020
        OP1_PTR=SUBSTR(AREA, 4, 4):                                     IFA01030
        OP2='TMP':                                                      IFA01040
        AREA=CXR:                                                       IFA01050
        AREA = TRANSLATE(AREA, ZERO0, BLANK):                           IFA01060
        OP2_PTR=SUBSTR(AREA, 4, 4):                                     IFA01070
        WRITE FILE (MATRIX) FROM (MTX_FIL):                             IFA01080
                                                                        IFA01090
    /* BRANCH TO ACTION 2 */                                            IFA01100
```

```
                                                                    IFA01110
     OPR='KEY':                                                     IFA01120
     OPR_PTR='0002':                                                IFA01130
     OP1='LIT':                                                     IFA01140
                                                                    IFA01150
     ORIGINAL_IDX = UST_IDX:                                        IFA01160
     DO WHILE ((US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23)):    IFA01170
          UST_IDX = UST_IDX + 1:                                    IFA01180
     END:                                                           IFA01190
                                                                    IFA01200
     AREA = UST_PTR(UST_IDX+1):                                     IFA01210
     AREA = TRANSLATE(AREA, ZERO0, BLANK):                          IFA01220
     OP1_PTR = SUBSTR(AREA, 4, 4):                                  IFA01230
                                                                    IFA01240
     OP2 = ' ':                                                     IFA01250
     OP2_PTR = '0000':                                              IFA01260
     WRITE FILE (MATRIX) FROM (MTX_FIL):                            IFA01270
                                                                    IFA01280
     UST_IDX = ORIGINAL_IDX:                                        IFA01290
                                                                    IFA01300
/* IS NEXT SYMBOL 'THEN' ? */                                       IFA01310
                                                                    IFA01320
     IF(US_NTRY(UST_IDX)¬='KEY')|(UST_PTR(UST_IDX)¬=5) THEN         IFA01330
          GO TO IF001:                                              IFA01340
                                                                    IFA01350
/* IS NEXT SYMBOL A KEYWORD ? */                                   IFA01360
                                                                    IFA01370
     UST_IDX = UST_IDX + 1:                                         IFA01380
     IF US_NTRY(UST_IDX)¬='KEY' THEN GO TO IF002:                   IFA01390
                                                                    IFA01400
/* CALL ACTION ROUTINE */                                          IFA01410
                                                                    IFA01420
     IF (UST_PTR(UST_IDX)=3)|                                       IFA01430
        (UST_PTR(UST_IDX)=4)|                                       IFA01440
        (UST_PTR(UST_IDX)=5)|                                       IFA01450
        (UST_PTR(UST_IDX)=6)|                                       IFA01460
        (UST_PTR(UST_IDX)=7)|                                       IFA01470
        (UST_PTR(UST_IDX)=10)|                                      IFA01480
        (UST_PTR(UST_IDX)=11)|                                      IFA01490
        (UST_PTR(UST_IDX)=12)|                                      IFA01500
        (UST_PTR(UST_IDX)=15)                                       IFA01510
        THEN DO:                                                    IFA01520
          ERROR(CNT)='INVALID ACTION SPECIFIED':                    IFA01530
          GO TO IF003:                                              IFA01540
        END:                                                        IFA01550
     IF UST_PTR(UST_IDX)=1 THEN DO:                                 IFA01560
          CALL LETAR:                                               IFA01570
          GO TO IF004:                                              IFA01580
          END:                                                      IFA01590
     IF UST_PTR(UST_IDX)=2 THEN DO:                                 IFA01600
          CALL GOTOAR:                                              IFA01610
          GO TO IF004:                                              IFA01620
          END:                                                      IFA01630
     IF UST_PTR(UST_IDX)=8 THEN DO:                                 IFA01640
          CALL GOSUBAR:                                             IFA01650
```

```
                GO TO IF004:                                              IFA01660
                END:                                                      IFA01670
            IF UST_PTR(UST_IDX)=9 THEN DO:                                IFA01680
                CALL RETRNAR:                                             IFA01690
                GO TO IF004:                                              IFA01700
                END:                                                      IFA01710
            IF UST_PTR(UST_IDX)=13 THEN DO:                               IFA01720
                CALL READAR:                                              IFA01730
                GO TO IF004:                                              IFA01740
                END:                                                      IFA01750
            IF UST_PTR(UST_IDX)=14 THEN DO:                               IFA01760
                CALL RESTRAR:                                             IFA01770
                GO TO IF004:                                              IFA01780
                END:                                                      IFA01790
            IF UST_PTR(UST_IDX)=16 THEN DO:                               IFA01800
                CALL STOPAR:                                              IFA01810
                GO TO IF004:                                              IFA01820
                END:                                                      IFA01830
            IF UST_PTR(UST_IDX)=17 THEN DO:                               IFA01840
                CALL ENDAR:                                               IFA01850
                GO TO IF004:                                              IFA01860
                END:                                                      IFA01870
            IF UST_PTR(UST_IDX)=18 THEN DO:                               IFA01880
                CALL WTOUTAR:                                             IFA01890
                GO TO IF004:                                              IFA01900
                END:                                                      IFA01910
            IF UST_PTR(UST_IDX)=19 THEN DO:                               IFA01920
                CALL RDINAR:                                              IFA01930
                GO TO IF004:                                              IFA01940
                END:                                                      IFA01950
                                                                          IFA01960
IF001:      /* IS SYMBOL A GOTO ? */                                      IFA01970
                                                                          IFA01980
            IF(US_NTRY(UST_IDX)='KEY')&(UST_PTR(UST_IDX)=2) THEN          IFA01990
                DO:                                                       IFA02000
                    CALL GOTOAR:                                          IFA02010
                    GO TO IF004:                                          IFA02020
                END:                                                      IFA02030
            ERROR(CNT)='THEN OR GOTO MUST FOLLOW CONDITION':              IFA02040
            GO TO IF003:                                                  IFA02050
                                                                          IFA02060
IF002:      /* IS SYMBOL A LINE NUMBER ? */                               IFA02070
                                                                          IFA02080
            IF(US_NTRY(UST_IDX)¬='LIT')|                                  IFA02090
                (VERIFY(LT_NTRY(UST_PTR(UST_IDX)),'0123456789')>7)|       IFA02100
                (VERIFY(LT_NTRY(UST_PTR(UST_IDX)),' 0123456789')¬=0)      IFA02110
                THEN DO:                                                  IFA02120
                    ERROR(CNT)='INVALID COMMAND FOLLOWS CONDITION':       IFA02130
                    GO TO IF003:                                          IFA02140
                END:                                                      IFA02150
            UST_IDX=UST_IDX-1:                                            IFA02160
            CALL GOTOAR:                                                  IFA02170
            GO TO IF004:                                                  IFA02180
                                                                          IFA02190
IF003:      /* ERROR ROUTINE */                                           IFA02200
```

```
FILE: IFAR      PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

                                                                                IFA02210
         SRS_LIN(CNT)=CRD_CTR;                                                  IFA02220
         CNT=CNT + 1;                                                           IFA02230
         DO WHILE ((US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23));           IFA02240
            UST_IDX=UST_IDX + 1;                                                IFA02250
         END;                                                                   IFA02260
         UST_IDX = UST_IDX + 1;                                                 IFA02270
         CRD_CTR = CRD_CTR + 1;                                                 IFA02280
         RETURN;                                                                IFA02290
                                                                                IFA02300
IF004:   /* NORMAL RETURN TO CONTROL */                                         IFA02310
   /* UST_IDX & CRD_CTR HAVE BEEN UPDATED IN ACTION ROUTINES */                 IFA02320
                                                                                IFA02330
         RETURN;                                                                IFA02340
                                                                                IFA02350
END IFAR;                                                                       IFA02360
```

```
FILE: CDX       PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

CDX : PROCEDURE:                                                          CDX00010
   DCL  1   UST                          (999)     EXT,                  CDX00020
            2   US_NTRY                  CHAR      (3),                  CDX00030
            2   UST_PTR                  FIXED     (4);                  CDX00040
   DCL  1   ERR_TBL                      (75)      EXT,                  CDX00050
            2   ERROR                    CHAR      (40),                 CDX00060
            2   SRS_LIN                  FIXED     (3);                  CDX00070
   DCL  1   TRM_TBL                      (30)      EXT,                  CDX00080
            2   TM_NTRY                  CHAR      (3),                  CDX00090
            2   TRM_PRI                  FIXED     (1);                  CDX00100
   DCL  1   MTX_FIL,                                                      CDX00110
            2   LBL_PTR                  CHAR      (4),                  CDX00120
            2   OPR                      CHAR      (3),                  CDX00130
            2   OPR_PTR                  CHAR      (4),                  CDX00140
            2   OP1                      CHAR      (3),                  CDX00150
            2   OP1_PTR                  CHAR      (4),                  CDX00160
            2   OP2                      CHAR      (3),                  CDX00170
            2   OP2_PTR                  CHAR      (4),                  CDX00180
            2   FILL                     CHAR      (55) INIT((55)' ');   CDX00190
   DCL  1   OPND                         CONTROLLED,                     CDX00200
            2   OPND_STACK_TBL           CHAR      (3),                  CDX00210
            2   OPND_STACK_PTR           FIXED     (4);                  CDX00220
   DCL  1   OPTR                         CONTROLLED,                     CDX00230
            2   OPTR_STACK_TBL           CHAR      (3),                  CDX00240
            2   OPTR_STACK_PTR           FIXED     (4),                  CDX00250
            2   OPTR_PRIORITY            FIXED     (4);                  CDX00260
   DCL  1   FCN_OPND                     CONTROLLED,                     CDX00270
            2   FCN_OPND_STACK_TBL       CHAR      (3),                  CDX00280
            2   FCN_OPND_STACK_PTR       FIXED     (4);                  CDX00290
   DCL  1   FCN_OPTR                     CONTROLLED,                     CDX00300
            2   FCN_OPTR_STACK_TBL       CHAR      (3),                  CDX00310
            2   FCN_OPTR_STACK_PTR       FIXED     (4),                  CDX00320
            2   FCN_OPTR_PRIORITY        FIXED     (4);                  CDX00330
   DCL  UST_IDX                          FIXED     (4)  EXT,             CDX00340
        CNT                              FIXED     (2)  EXT,             CDX00350
        AREA                             CHAR      (7),                  CDX00360
        ZERO0                            CHAR      (1)  INIT('0'),       CDX00370
        BLANK                            CHAR      (1)  INIT(' '),       CDX00380
        CRD_CTR                          FIXED     (3)  EXT,             CDX00390
        TGT_PTR                          FIXED     (4)  EXT,             CDX00400
        TMP_STR                          FIXED     (3)  EXT,             CDX00410
        NO_FLG                           FIXED     (1),                  CDX00420
        PAR_CT                           FIXED     (3),                  CDX00430
        PRIORITY                         FIXED     (3),                  CDX00440
        FCN_PAR_CT                       FIXED     (3),                  CDX00450
        FCN_FLG                          FIXED     (1),                  CDX00460
        ASGN                             FIXED     (4),                  CDX00470
        FCN_PTR                          FIXED     (4);                  CDX00480
   DCL  MATRIX                           FILE      RECORD;               CDX00490
                                                                          CDX00500
   NO_FLG, PAR_CT, PRIORITY, FCN_PAR_CT, FCN_FLG = 0;                     CDX00510
                                                                          CDX00520
   /* CHECK FOR LEFT PARENTHESIS, RIGHT PARENTHESIS, LINE END */          CDX00530
                                                                          CDX00540
LA002:  UST_IDX = UST_IDX + 1;                                            CDX00550
```

```
FILE: CDX       PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

          IF(US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=7) THEN                CDX00560
               DO:                                                            CDX00570
                    IF FCN_FLG = 1 THEN FCN_PAR_CT = FCN_PAR_CT + 10:         CDX00580
                    PAR_CT = PAR_CT + 10:                                     CDX00590
                    GO TO LA002:                                              CDX00600
               END:                                                           CDX00610
LA003:    IF(US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=8) THEN                CDX00620
               DO:                                                            CDX00630
                    IF FCN_FLG=1 THEN FCN_PAR_CT = FCN_PAR_CT-10:             CDX00640
                    PAR_CT = PAR_CT - 10:                                     CDX00650
                    UST_IDX = UST_IDX + 1:                                    CDX00660
                    GO TO LA003:                                              CDX00670
               END:                                                           CDX00680
          IF((US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=6))|                  CDX00690
               ((US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=9))|               CDX00700
               ((US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=10))|              CDX00710
               ((US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=24))|              CDX00720
               ((US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=25))|              CDX00730
               ((US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=26))|              CDX00740
               ((US_NTRY(UST_IDX)='KEY')&(UST_PTR(UST_IDX)=2))|               CDX00750
               ((US_NTRY(UST_IDX)='KEY')&(UST_PTR(UST_IDX)=5)) THEN           CDX00760
               DO:                                                            CDX00770
                    NO_FLG = 1:                                               CDX00780
                    GO TO LA004_POP:                                          CDX00790
               END:                                                           CDX00800
                                                                              CDX00810
     /* IS SYMBOL A FUNCTION NAME ? */                                        CDX00820
                                                                              CDX00830
          IF(US_NTRY(UST_IDX) ='TRM')&(UST_PTR(UST_IDX)>12) &                 CDX00840
               (UST_PTR(UST_IDX)<23) THEN                                     CDX00850
               DO:                                                            CDX00860
                    FCN_FLG = 1:                                              CDX00870
                    FCN_PTR = UST_PTR(UST_IDX):                               CDX00880
                    UST_IDX = UST_IDX + 1:                                    CDX00890
                    IF(US_NTRY(UST_IDX)¬='TRM')|                              CDX00900
                         (UST_PTR(UST_IDX)¬=7) THEN                           CDX00910
                         DO:                                                  CDX00920
               ERROR(CNT)='PARENTHESIS MUST ENCLOSE FUNCTION ARG':            CDX00930
                              GO TO LA001_ERROR:                              CDX00940
                         END:                                                 CDX00950
                    UST_IDX = UST_IDX - 1:                                    CDX00960
                    GO TO LA002:                                              CDX00970
               END:                                                           CDX00980
                                                                              CDX00990
     /* IS SYMBOL A LITERAL OR IDENTIFIER ? */                                CDX01000
                                                                              CDX01010
          IF(US_NTRY(UST_IDX)¬='LIT')&(US_NTRY(UST_IDX)¬='IDN') THEN          CDX01020
               DO:                                                            CDX01030
                    ERROR(CNT)='INVALID SYNTAX':                              CDX01040
                    GO TO LA001_ERROR:                                        CDX01050
               END:                                                           CDX01060
                                                                              CDX01070
     /* PUSH SYMBOL ONTO APPROPRIATE STACK */                                 CDX01080
                                                                              CDX01090
          IF FCN_FLG = 1 THEN                                                 CDX01100
```

```
FILE: CDX       PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

                DO:                                                     CDX01110
                    ALLOCATE FCN_OPND:                                  CDX01120
                    FCN_OPND_STACK_TAL = US_NTRY(UST_IDX):              CDX01130
                    FCN_OPND_STACK_PTR = UST_PTR(UST_IDX):              CDX01140
                    GO TO LA005:                                        CDX01150
                END:                                                    CDX01160
            ALLOCATE OPND:                                              CDX01170
            OPND_STACK_TAL = US_NTRY(UST_IDX):                          CDX01180
            OPND_STACK_PTR = UST_PTR(UST_IDX):                          CDX01190
                                                                        CDX01200
        /* CHECK FOR LEFT PARENTHESIS, RIGHT PARENTHESIS, LINE END */   CDX01210
                                                                        CDX01220
LA005:  UST_IDX = UST_IDX + 1:                                          CDX01230
            IF(US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=7) THEN        CDX01240
                DO:                                                     CDX01250
                    IF FCN_FLG = 1 THEN FCN_PAR_CT = FCN_PAR_CT+10:     CDX01260
                    PAR_CT = PAR_CT + 10:                               CDX01270
                    GO TO LA005:                                        CDX01280
                END:                                                    CDX01290
LA006:      IF(US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=8) THEN        CDX01300
                DO:                                                     CDX01310
                    IF FCN_FLG = 1 THEN FCN_PAR_CT = FCN_PAR_CT-10:     CDX01320
                    PAR_CT = PAR_CT -10:                                CDX01330
                    IF(FCN_PAR_CT=0)&(FCN_FLG=1) THEN                   CDX01340
                        DO:                                             CDX01350
                            IF(ALLOCATION(FCN_OPTR)=0) THEN             CDX01360
                                GO TO LA007_POP_FCN:                    CDX01370
                            GO TO LA004_POP:                            CDX01380
                        END:                                            CDX01390
                    UST_IDX = UST_IDX + 1:                              CDX01400
                    GO TO LA006:                                        CDX01410
                END:                                                    CDX01420
            IF((US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=6))|          CDX01430
                ((US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=9))|        CDX01440
                ((US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=10))|       CDX01450
                ((US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=24))|       CDX01460
                ((US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=25))|       CDX01470
                ((US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=26))|       CDX01480
                ((US_NTRY(UST_IDX)='KEY')&(UST_PTR(UST_IDX)=2))|        CDX01490
                ((US_NTRY(UST_IDX)='KEY')&(UST_PTR(UST_IDX)=5)) THEN    CDX01500
                DO:                                                     CDX01510
                    NO_FLG = 1:                                         CDX01520
                    IF ALLOCATION(OPTR)=0 THEN GO TO LA009_FINISH:      CDX01530
                    GO TO LA004_POP:                                    CDX01540
                END:                                                    CDX01550
                                                                        CDX01560
        /* IS SYMBOL AN OPERATOR ? */                                   CDX01570
            IF(US_NTRY(UST_IDX)^='TRM')|(UST_PTR(UST_IDX)>5) THEN       CDX01580
                DO:                                                     CDX01590
                    ERROR(CNT)='INVALID SYNTAX':                        CDX01600
                    GO TO LA001_ERROR:                                  CDX01610
                END:                                                    CDX01620
            PRIORITY=TRM_PRI(UST_PTR(UST_IDX))+PAR_CT:                  CDX01630
                                                                        CDX01640
        /* IS OPERATOR STACK EMPTY ? */                                 CDX01650
```

```
        IF FCN_FLG = 1 THEN                                          CDX01660
            DO:                                                      CDX01670
                IF ALLOCATION(FCN_OPTR)=0 THEN                       CDX01680
                    GO TO LA008_PUSH_OPTR:                           CDX01690
                GO TO LA010:                                         CDX01700
            END:                                                     CDX01710
        IF ALLOCATION(OPTR)=0 THEN GO TO LA008_PUSH_OPTR:            CDX01720
                                                                     CDX01730
    /* IS TOP OF STACK GREATER THAN PRIORITY ? */                    CDX01740
                                                                     CDX01750
LA010:  IF FCN_FLG = 1 THEN                                          CDX01760
            DO:                                                      CDX01770
                IF FCN_OPTR_PRIORITY > PRIORITY THEN                 CDX01780
                    GO TO LA004_POP:                                 CDX01790
                GO TO LA008_PUSH_OPTR:                               CDX01800
            END:                                                     CDX01810
        IF OPTR_PRIORITY > PRIORITY THEN                             CDX01820
            GO TO LA004_POP:                                         CDX01830
        GO TO LA008_PUSH_OPTR:                                       CDX01840
                                                                     CDX01850
    /* POP STACKS AND WRITE INTO MATRIX */                           CDX01860
                                                                     CDX01870
LA004_POP:                                                           CDX01880
        IF FCN_FLG = 1 THEN                                          CDX01890
            DO:                                                      CDX01900
                AREA=TGT_PTR:                                        CDX01910
                AREA=TRANSLATE(AREA, ZERO0, BLANK):                  CDX01920
                LBL_PTR=SUBSTR(AREA, 4, 4):                          CDX01930
                OPR=FCN_OPTR_STACK_TBL:                              CDX01940
                AREA = FCN_OPTR_STACK_PTR:                           CDX01950
                AREA=TRANSLATE(AREA, ZERO0, BLANK):                  CDX01960
                OPR_PTR = SUBSTR(AREA, 4, 4):                        CDX01970
                FREE FCN_OPTR:                                       CDX01980
                OP2=FCN_OPND_STACK_TBL:                              CDX01990
                AREA=FCN_OPND_STACK_PTR:                             CDX02000
                AREA=TRANSLATE(AREA, ZERO0, BLANK):                  CDX02010
                OP2_PTR = SUBSTR(AREA, 4, 4):                        CDX02020
                FREE FCN_OPND:                                       CDX02030
                OP1=FCN_OPND_STACK_TBL:                              CDX02040
                AREA=FCN_OPND_STACK_PTR:                             CDX02050
                AREA=TRANSLATE(AREA, ZERO0, BLANK):                  CDX02060
                OP1_PTR=SUBSTR(AREA, 4, 4):                          CDX02070
                FREE FCN_OPND:                                       CDX02080
                WRITE FILE (MATRIX) FROM (MTX_FIL):                  CDX02090
                GO TO LA011:                                         CDX02100
            END:                                                     CDX02110
        AREA=TGT_PTR:                                                CDX02120
        AREA=TRANSLATE(AREA, ZERO0, BLANK):                          CDX02130
        LBL_PTR=SUBSTR(AREA, 4, 4):                                  CDX02140
        OPR=OPTR_STACK_TBL:                                          CDX02150
        AREA=OPTR_STACK_PTR:                                         CDX02160
        AREA=TRANSLATE(AREA, ZERO0, BLANK):                          CDX02170
        OPR_PTR=SUBSTR(AREA, 4, 4):                                  CDX02180
        FREE OPTR:                                                   CDX02190
        OP2=OPND_STACK_TBL:                                          CDX02200
```

```
FILE: CDX       PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

          AREA=OPND_STACK_PTR:                                          CDX02210
          AREA=TRANSLATE(AREA, ZERO0, BLANK):                           CDX02220
          OP2_PTR=SUBSTR(AREA, 4, 4):                                   CDX02230
          FREE OPND:                                                    CDX02240
          OP1=OPND_STACK_TBL:                                           CDX02250
          AREA=OPND_STACK_PTR:                                          CDX02260
          AREA=TRANSLATE(AREA, ZERO0, BLANK):                           CDX02270
          OP1_PTR=SUBSTR(AREA, 4, 4):                                   CDX02280
          FREE OPND:                                                    CDX02290
          WRITE FILE (MATRIX) FROM (MTX_FIL):                           CDX02300
                                                                        CDX02310
     /* ASSIGN TEMPORARY VALUE AND PUSH ONTO APPROPRIATE STACK*/        CDX02320
                                                                        CDX02330
LA011:TMP_STR=TMP_STR + 1;                                              CDX02340
          IF FCN_FLG = 1 THEN                                           CDX02350
               DO:                                                      CDX02360
                    ALLOCATE FCN_OPND:                                  CDX02370
                    FCN_OPND_STACK_TBL = 'TMP':                         CDX02380
                    FCN_OPND_STACK_PTR=TMP_STR:                         CDX02390
                    GO TO LA012:                                        CDX02400
               END:                                                     CDX02410
          ALLOCATE OPND:                                                CDX02420
          OPND_STACK_TBL = 'TMP':                                       CDX02430
          OPND_STACK_PTR=TMP_STR:                                       CDX02440
                                                                        CDX02450
     /* CHECK FOR END OF FUNCTION */                                    CDX02460
                                                                        CDX02470
LA012:IF(FCN_PAR_CT=0)&(FCN_FLG=1) THEN                                 CDX02480
               DO:                                                      CDX02490
                    IF ALLOCATION(FCN_OPTR)=0 THEN                      CDX02500
                         GO TO LA007_POP_FCN:                           CDX02510
                    GO TO LA004_POP:                                    CDX02520
               END:                                                     CDX02530
                                                                        CDX02540
     /* IS OPERATOR STACK EMPTY AND END FLAG SET ? */                   CDX02550
                                                                        CDX02560
          IF (ALLOCATION(OPTR)=0)&(ND_FLG=1) THEN                       CDX02570
               GO TO LA009_FINISH:                                      CDX02580
                                                                        CDX02590
     /* CHECK FOR END OF LINE */                                        CDX02600
                                                                        CDX02610
          IF(ND_FLG=1) THEN                                             CDX02620
               GO TO LA004_POP:                                         CDX02630
                                                                        CDX02640
     /* CHECK FOR EMPTY OPERATOR STACK */                               CDX02650
                                                                        CDX02660
          IF FCN_FLG = 1 THEN                                           CDX02670
               DO:                                                      CDX02680
                    IF ALLOCATION(FCN_OPTR)=0 THEN                      CDX02690
                         GO TO LA008_PUSH_OPTR:                         CDX02700
                    GO TO LA010:                                        CDX02710
               END:                                                     CDX02720
          IF ALLOCATION(OPTR)=0 THEN                                    CDX02730
               GO TO LA008_PUSH_OPTR:                                   CDX02740
          GO TO LA010:                                                  CDX02750
```

```
FILE: CDX      PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

                                                                        CDX02760
    /* PUSH OPERATOR ONTO APPROPRIATE STACK */                          CDX02770
                                                                        CDX02780
LA008_PUSH_OPTR:                                                        CDX02790
        IF FCN_FLG = 1 THEN                                             CDX02800
            DO;                                                         CDX02810
                ALLOCATE FCN_OPTR;                                      CDX02820
                FCN_OPTR_STACK_TRL = US_NTRY(UST_IDX);                  CDX02830
                FCN_OPTR_STACK_PTR = UST_PTR(UST_IDX);                  CDX02840
                FCN_OPTR_PRIORITY = PRIORITY;                           CDX02850
                GO TO LA002;                                            CDX02860
            END;                                                        CDX02870
        ALLOCATE OPTR;                                                  CDX02880
        OPTR_STACK_TRL = US_NTRY(UST_IDX);                              CDX02890
        OPTR_STACK_PTR = UST_PTR(UST_IDX);                              CDX02900
        OPTR_PRIORITY = PRIORITY;                                       CDX02910
        GO TO LA002;                                                    CDX02920
                                                                        CDX02930
    /* FINISH FUNCTION PROCESSING */                                    CDX02940
                                                                        CDX02950
LA007_POP_FCN:                                                          CDX02960
        AREA=TGT_PTR;                                                   CDX02970
        AREA=TRANSLATE(AREA, ZERO0, BLANK);                             CDX02980
        LBL_PTR = SUBSTR (AREA, 4, 4);                                  CDX02990
        OPR='TRM';                                                      CDX03000
        AREA = FCN_PTR;                                                 CDX03010
        AREA=TRANSLATE(AREA, ZERO0, BLANK);                             CDX03020
        OPR_PTR=SUBSTR(AREA, 4, 4);                                     CDX03030
        OP1=FCN_OPND_STACK_TRL;                                         CDX03040
        AREA=FCN_OPND_STACK_PTR;                                        CDX03050
        AREA=TRANSLATE(AREA, ZERO0, BLANK);                             CDX03060
        OP1_PTR=SUBSTR(AREA, 4, 4);                                     CDX03070
        FREE FCN_OPND;                                                  CDX03080
        OP2='    ';                                                     CDX03090
        OP2_PTR='0000';                                                 CDX03100
        WRITE FILE (MATRIX) FROM (MTX_FIL);                             CDX03110
        FCN_FLG=0;                                                      CDX03120
        TMP_STR=TMP_STR + 1;                                            CDX03130
        ALLOCATE OPND;                                                  CDX03140
        OPND_STACK_TRL = 'TMP';                                         CDX03150
        OPND_STACK_PTR = TMP_STR;                                       CDX03160
        GO TO LA005;                                                    CDX03170
                                                                        CDX03180
    /* FINISH ASSIGNMENT STATEMENT AND RETURN TO CONTROL */             CDX03190
                                                                        CDX03200
LA009_FINISH:                                                           CDX03210
                                                                        CDX03220
    /* CHECK FOR AGREEMENT OF PARENTHESES */                            CDX03230
                                                                        CDX03240
        IF PAR_CT¬=0 THEN                                               CDX03250
            DO;                                                         CDX03260
                ERROR(CNT)='UNEQUAL NO. OF LEFT AND RIGHT PAREN.';      CDX03270
                GO TO LA001_ERROR;                                      CDX03280
            END;                                                        CDX03290
        AREA=TGT_PTR;                                                   CDX03300
```

```
FILE: CDX       PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

        AREA=TRANSLATE(AREA, ZERO0, BLANK);                                  CDX03310
        L3L_PTR=SUBSTR(AREA, 4, 4);                                          CDX03320
        OPR='TRM';                                                           CDX03330
        OPR_PTR='0006';                                                      CDX03340
        OP2=OPND_STACK_TAL;                                                  CDX03350
        AREA=OPND_STACK_PTR;                                                 CDX03360
        AREA=TRANSLATE(AREA, ZERO0, BLANK);                                  CDX03370
        OP2_PTR=SUBSTR(AREA, 4, 4);                                          CDX03380
        FREE OPND;                                                           CDX03390
        OP1='TMP';                                                           CDX03400
        TMP_STR=TMP_STR + 1;                                                 CDX03410
        AREA=TMP_STR;                                                        CDX03420
        AREA=TRANSLATE(AREA, ZERO0, BLANK);                                  CDX03430
        OP1_PTR=SUBSTR(AREA, 3, 4);                                          CDX03440
        WRITE FILE (MATRIX) FROM (MTX_FIL);                                  CDX03450
        RETURN;                                                              CDX03460
                                                                             CDX03470
    /* ERROR ROUTINE */                                                      CDX03480
                                                                             CDX03490
LA001_ERROR:                                                                 CDX03500
        SRS_LIN(CNT)=CRD_CTR;                                                CDX03510
        CNT=CNT+1;                                                           CDX03520
        DO WHILE ((UST_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23));        CDX03530
            UST_IDX = UST_IDX + 1;                                           CDX03540
        END;                                                                 CDX03550
        UST_IDX = UST_IDX + 1;                                               CDX03560
        CRD_CTR = CRD_CTR + 1;                                               CDX03570
                                                                             CDX03580
    /* CLEAR STACKS */                                                       CDX03590
                                                                             CDX03600
        DO WHILE(ALLOCATION(OPND)¬=0);                                       CDX03610
            FREE OPND;                                                       CDX03620
        END;                                                                 CDX03630
        DO WHILE(ALLOCATION(OPTR)¬=0);                                       CDX03640
            FREE OPTR;                                                       CDX03650
        END;                                                                 CDX03660
        DO WHILE(ALLOCATION(ECN_OPND)¬=0);                                   CDX03670
            FREE ECN_OPND;                                                   CDX03680
        END;                                                                 CDX03690
        DO WHILE(ALLOCATION(ECN_OPTR)¬=0);                                   CDX03700
            FREE ECN_OPTR;                                                   CDX03710
        END;                                                                 CDX03720
        RETURN;                                                              CDX03730
                                                                             CDX03740
END CDX;                                                                     CDX03750
```

```
FILE: DATAAR   PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

 DATAAR : PROCEDURE:                                                          DAT00010
   DCL  1   UST                          (999)     EXT,                     DAT00020
            2   US_NTRY                  CHAR      (3),                     DAT00030
            2   UST_PTR                  FIXED     (4);                     DAT00040
   DCL  1   ERR_TBL                      (75)      EXT,                     DAT00050
            2   ERROR                    CHAR      (40),                    DAT00060
            2   SRS_LIN                  FIXED     (3);                     DAT00070
   DCL  1   DTA_TBL                      (101)     EXT,                     DAT00080
            2   DT_PTR                   FIXED     (3);                     DAT00090
   DCL  TGT_PTR                          FIXED     (4)  EXT,                DAT00100
        DT_IDX                           FIXED     (3)  EXT,                DAT00110
        UST_IDX                          FIXED     (4)  EXT,                DAT00120
        CNT                              FIXED     (2)  EXT,                DAT00130
        CRD_CTR                          FIXED     (3)  EXT;                DAT00140
                                                                            DAT00150
   /* IS THE NEXT UNIFORM SYMBOL A VALID LITERAL ? */                        DAT00160
                                                                            DAT00170
        UST_IDX = UST_IDX + 1;                                              DAT00180
   DA001:IF US_NTRY(UST_IDX)='LIT' THEN                                     DAT00190
            DO;                                                             DAT00200
               IF DT_IDX > 100 THEN GO TO DA004;                            DAT00210
               DT_PTR(DT_IDX)=UST_PTR(UST_IDX);                             DAT00220
               DT_IDX=DT_IDX + 1;                                           DAT00230
               UST_IDX = UST_IDX + 1;                                       DAT00240
               GO TO DA002;                                                 DAT00250
            END;                                                            DAT00260
        ERROR(CNT)='DATA ENTRIES MUST BE NUMERIC';                          DAT00270
        SRS_LIN(CNT)=CRD_CTR;                                               DAT00280
        CNT=CNT+1;                                                          DAT00290
        DO WHILE((US_NTRY(UST_IDX) ¬='TRM')|(UST_PTR(UST_IDX)¬=23));        DAT00300
            UST_IDX = UST_IDX + 1;                                          DAT00310
        END;                                                                DAT00320
        UST_IDX=UST_IDX + 1;                                                DAT00330
        CRD_CTR = CRD_CTR + 1;                                              DAT00340
        RETURN;                                                             DAT00350
   DA002:IF(US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=11) THEN              DAT00360
                                                                            DAT00370
   /* NEXT SYMBOL IS A COMMA */                                             DAT00380
                                                                            DAT00390
        DO;                                                                 DAT00400
            UST_IDX = UST_IDX + 1;                                          DAT00410
            GO TO DA001;                                                    DAT00420
        END;                                                                DAT00430
                                                                            DAT00440
   /* IS NEXT UNIFORM SYMBOL A $ ? */                                       DAT00450
                                                                            DAT00460
        IF(US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=23) THEN               DAT00470
            DO;                                                             DAT00480
                                                                            DAT00490
   /* NORMAL RETURN TO CONTROL */                                           DAT00500
                                                                            DAT00510
               UST_IDX = UST_IDX + 1;                                       DAT00520
               CRD_CTR =CRD_CTR + 1;                                        DAT00530
               RETURN;                                                      DAT00540
            END;                                                            DAT00550
```

```
FILE: DATAAR   PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

                                                                        DAT00560
    /* ERROR */                                                         DAT00570
                                                                        DAT00580
        ERROR(CNT) = 'COMMAS REQUIRED BETWEEN DATA VALUES';             DAT00590
        SRS_LIN(CNT)=CRD_CTR;                                           DAT00600
        CNT=CNT+1;                                                      DAT00610
        DO WHILE((US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23));     DAT00620
            UST_IDX = UST_IDX + 1;                                      DAT00630
        END;                                                            DAT00640
        UST_IDX = UST_IDX + 1;                                          DAT00650
        CRD_CTR = CRD_CTR + 1;                                          DAT00660
        RETURN;                                                         DAT00670
                                                                        DAT00680
    /* STORAGE FOR DATA EXCEEDED */                                     DAT00690
                                                                        DAT00700
    DA004:ERROR(CNT)='DATA ENTRIES EXCEEDS CAPACITY OF 100';            DAT00710
        SRS_LIN(CNT)=CRD_CTR;                                           DAT00720
        CNT=CNT+1;                                                      DAT00730
        DO WHILE((US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23));     DAT00740
            UST_IDX = UST_IDX + 1;                                      DAT00750
        END;                                                            DAT00760
        UST_IDX = UST_IDX + 1;                                          DAT00770
        CRD_CTR = CRD_CTR + 1;                                          DAT00780
        RETURN;                                                         DAT00790
END DATAAR;                                                             DAT00800
```

```
FILE: RETRNAR  PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

 RETRNAR : PROCEDURE:                                                    RET00010
    DCL  1    UST                          (999)      EXT,              RET00020
              2    US_NTRY                 CHAR       (3),              RET00030
              2    UST_PTR                 FIXED      (4):              RET00040
    DCL  1    ERR_TAL                      (75)       EXT,              RET00050
              2    ERROR                   CHAR       (40),             RET00060
              2    SRS_LIN                 FIXED      (3):              RET00070
    DCL  1    MTX_FIL,                                                  RET00080
              2    LBL_PTR                 CHAR       (4),              RET00090
              2    OPR                     CHAR       (3),              RET00100
              2    OPR_PTR                 CHAR       (4),              RET00110
              2    OP1                     CHAR       (3),              RET00120
              2    OP1_PTR                 CHAR       (4),              RET00130
              2    OP2                     CHAR       (3),              RET00140
              2    OP2_PTR                 CHAR       (4),              RET00150
              2    FILL                    CHAR       (55) INIT((55)' '): RET00160
    DCL  UST_IDX                           FIXED      (4)  EXT,         RET00170
         CNT                               FIXED      (2)  EXT,         RET00180
         CRD_CTR                           FIXED      (3)  EXT,         RET00190
         AREA                              CHAR       (7),              RET00200
         ZERO0                             CHAR       (1)  INIT('0'),   RET00210
         BLANK                             CHAR       (1)  INIT(' '),   RET00220
         TGT_PTR                           FIXED      (4)  EXT:         RET00230
    DCL  MATRIX                            FILE       RECORD:           RET00240
                                                                         RET00250
    /* THE NEXT UNIFORM SYMBOL MUST BE '$' */                            RET00260
                                                                         RET00270
         UST_IDX = UST_IDX + 1:                                          RET00280
         IF(US_NTRY(UST_IDX) ¬= 'TRM')|(UST_PTR(UST_IDX)¬=23) THEN       RET00290
              GO TO RT001:                                               RET00300
         AREA = TGT_PTR:                                                 RET00310
         AREA = TRANSLATE (AREA, ZERO0, BLANK):                          RET00320
         LBL_PTR = SUBSTR (AREA, 4, 4):                                  RET00330
         OPR = 'KEY':                                                    RET00340
         OPR_PTR = '0009':                                               RET00350
         OP1 = '   ':                                                   RET00360
         OP1_PTR = '0000':                                               RET00370
         OP2 = '   ':                                                   RET00380
         OP2_PTR = '0000':                                               RET00390
         WRITE FILE (MATRIX) FROM (MTX_FIL):                             RET00400
         UST_IDX = UST_IDX + 1:                                          RET00410
         CRD_CTR = CRD_CTR + 1:                                          RET00420
         RETURN:                                                         RET00430
    RT001:ERROR(CNT) = 'CHARACTERS APPEAR AFTER RETURN STMT':            RET00440
         SRS_LIN(CNT) = CRD_CTR:                                         RET00450
         CNT = CNT + 1:                                                  RET00460
         DO WHILE((US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23)):     RET00470
              UST_IDX = UST_IDX + 1:                                     RET00480
         END:                                                            RET00490
         UST_IDX = UST_IDX + 1:                                          RET00500
         CRD_CTR = CRD_CTR + 1:                                          RET00510
         RETURN:                                                         RET00520
END RETRNAR:                                                             RET00530
```

```
FILE: STOPAR   PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

 STOPAR : PROCEDURE:                                                          ST000010
    DCL  1    UST                          (999)     EXT,                   ST000020
              2   US_NTRY                  CHAR      (3),                   ST000030
              2   UST_PTR                  FIXED     (4);                   ST000040
    DCL  1    ERR_TBL                      (75)      EXT,                   ST000050
              2   ERROR                    CHAR      (40),                  ST000060
              2   SRS_LIN                  FIXED     (3);                   ST000070
    DCL  1    MTX_FIL,                                                       ST000080
              2   LAL_PTR                  CHAR      (4),                   ST000090
              2   OPR                      CHAR      (3),                   ST000100
              2   OPR_PTR                  CHAR      (4),                   ST000110
              2   OP1                      CHAR      (3),                   ST000120
              2   OP1_PTR                  CHAR      (4),                   ST000130
              2   OP2                      CHAR      (3),                   ST000140
              2   OP2_PTR                  CHAR      (4),                   ST000150
              2   FILL                     CHAR      (55) INIT((55)' '):    ST000160
    DCL  UST_IDX                           FIXED     (4)  EXT,              ST000170
         CNT                               FIXED     (2)  EXT,              ST000180
         AREA                              CHAR      (7),                   ST000190
         ZERO0                             CHAR      (1)  INIT('0'),        ST000200
         BLANK                             CHAR      (1)  INIT(' '),        ST000210
         CRD_CTR                           FIXED     (3)  EXT,              ST000220
         TGT_PTR                           FIXED     (4)  EXT;              ST000230
    DCL  MATRIX                            FILE      RECORD;                ST000240
                                                                            ST000250
    /* THE NEXT UNIFORM SYMBOL MUST BE 'S' */                               ST000260
                                                                            ST000270
         UST_IDX = UST_IDX + 1;                                             ST000280
         IF(US_NTRY(UST_IDX) ¬= 'TRM')|(UST_PTR(UST_IDX)¬=23) THEN          ST000290
              GO TO SA001;                                                  ST000300
         AREA = TGT_PTR;                                                    ST000310
         AREA = TRANSLATE (AREA, ZERO0, BLANK);                             ST000320
         LAL_PTR = SUBSTR (AREA, 4, 4);                                     ST000330
         OPR = 'KEY';                                                       ST000340
         OPR_PTR = '0016';                                                  ST000350
         OP1 = '  Y';                                                       ST000360
         OP1_PTR = '0000';                                                  ST000370
         OP2 = '   ';                                                       ST000380
         OP2_PTR = '0000';                                                  ST000390
         WRITE FILE (MATRIX) FROM (MTX_FIL);                                ST000400
         UST_IDX = UST_IDX + 1;                                             ST000410
         CRD_CTR = CRD_CTR + 1;                                             ST000420
         RETURN;                                                            ST000430
    SA001:ERROR(CNT) = 'CHARACTERS APPEAR AFTER STOP STMT';                 ST000440
         SRS_LIN(CNT) = CRD_CTR;                                            ST000450
         CNT = CNT + 1;                                                     ST000460
         DO WHILE((US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23));        ST000470
              UST_IDX = UST_IDX + 1;                                        ST000480
         END;                                                               ST000490
         UST_IDX = UST_IDX + 1;                                             ST000500
         CRD_CTR = CRD_CTR + 1;                                             ST000510
         RETURN;                                                            ST000520
 END STOPAR;                                                                ST000530
```

```
FILE: WTOUTAR  PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

  WTOUTAR : PROCEDURE:                                                    WT000010
                                                                          WT000020
    DCL   1   UST                       (999)     EXT,                  WT000030
              2   US_NTRY               CHAR      (3),                  WT000040
              2   UST_PTR               FIXED     (4);                  WT000050
    DCL   1   ERR_TBL                   (75)      EXT,                  WT000060
              2   ERROR                 CHAR      (40),                 WT000070
              2   SRS_LIN               FIXED     (3);                  WT000080
    DCL   1   LIT_TBL                   (500)     EXT,                  WT000090
              2   LT_NTRY               CHAR      (9);                  WT000100
    DCL   1   MTX_FIL,                                                    WT000110
              2   LBL_PTR               CHAR      (4),                  WT000120
              2   OPR                   CHAR      (3),                  WT000130
              2   OPR_PTR               CHAR      (4),                  WT000140
              2   OP1                   CHAR      (3),                  WT000150
              2   OP1_PTR               CHAR      (4),                  WT000160
              2   OP2                   CHAR      (3),                  WT000170
              2   OP2_PTR               CHAR      (4),                  WT000180
              2   FILL                  CHAR      (55) INIT((55)' '):   WT000190
    DCL   UST_IDX                       FIXED     (4)  EXT,             WT000200
          AREA                          CHAR      (7),                  WT000210
          ZERO0                         CHAR      (1)  INIT('0'),       WT000220
          BLANK                         CHAR      (1)  INIT(' '),       WT000230
          CNT                           FIXED     (2)  EXT,             WT000240
          CRD_CTR                       FIXED     (3)  EXT,             WT000250
          TGT_PTR                       FIXED     (4)  EXT;             WT000260
    DCL   MATRIX                        FILE      RECORD;               WT000270
                                                                          WT000280
     /* IS NEXT UNIFORM SYMBOL AN IDENTIFIER ? */                         WT000290
                                                                          WT000300
          UST_IDX = UST_IDX + 1;                                          WT000310
          IF US_NTRY(UST_IDX)¬='IDN' THEN                                 WT000320
               DO;                                                        WT000330
                    ERROR(CNT)='VARIABLE NAME MUST FOLLOW WTOUT';         WT000340
                    SRS_LIN(CNT)=CRD_CTR;                                 WT000350
                    CNT=CNT+1;                                            WT000360
                    DO WHILE((US_NTRY(UST_IDX)¬='TRM')|                   WT000370
                         (UST_PTR(UST_IDX)¬=23));                         WT000380
                         UST_IDX = UST_IDX + 1;                           WT000390
                    END;                                                  WT000400
                    UST_IDX = UST_IDX + 1;                                WT000410
                    CRD_CTR = CRD_CTR + 1;                                WT000420
                    RETURN;                                               WT000430
               END;                                                       WT000440
                                                                          WT000450
     /* WRITE IDENTIFIER INTO MATRIX */                                   WT000460
                                                                          WT000470
          AREA = TGT_PTR;                                                 WT000480
          AREA = TRANSLATE (AREA, ZERO0, BLANK);                          WT000490
          LBL_PTR = SUBSTR (AREA, 4, 4);                                  WT000500
          OPR = 'KEY';                                                    WT000510
          OPR_PTR = '0018';                                               WT000520
          OP1 = 'IDN';                                                    WT000530
          AREA = UST_PTR(UST_IDX);                                        WT000540
          AREA = TRANSLATE (AREA, ZERO0, BLANK);                          WT000550
```

FILE: WTOUTAR PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
        OP1_PTR = SUBSTR (AREA, 4, 4);                                  WT000560
                                                                        WT000570
    /* IS NEXT UNIFORM SYMBOL 'TO' ? */                                 WT000580
                                                                        WT000590
        UST_IDX = UST_IDX + 1;                                          WT000600
        IF(US_NTRY(UST_IDX)¬='KEY')|(UST_PTR (UST_IDX)¬=4) THEN          WT000610
            DO;                                                         WT000620
                ERROR(CNT)='TO MUST FOLLOW VARIABLE NAME';              WT000630
                SRS_LIN(CNT) = CRD_CTR;                                 WT000640
                CNT = CNT + 1;                                          WT000650
                DO WHILE ((US_NTRY(UST_IDX)¬='TRM')|                    WT000660
                     (UST_PTR(UST_IDX)¬=23));                           WT000670
                     UST_IDX = UST_IDX + 1;                             WT000680
                END;                                                    WT000690
                UST_IDX = UST_IDX + 1;                                  WT000700
                CRD_CTR = CRD_CTR + 1;                                  WT000710
                RETURN;                                                 WT000720
            END;                                                        WT000730
                                                                        WT000740
    /* IS NEXT UNIFORM SYMBOL A HEX ADDRESS ? */                        WT000750
                                                                        WT000760
        UST_IDX=UST_IDX+1;                                              WT000770
        IF(US_NTRY(UST_IDX)¬='LIT')|                                    WT000780
            (INDEX(LT_NTRY(UST_PTR(UST_IDX)),',')¬=0)|                  WT000790
            (INDEX(LT_NTRY(UST_PTR(UST_IDX)),' ')¬=5)                   WT000800
            THEN DO;                                                    WT000810
                ERROR(CNT)='HEX ADDRESS MUST FOLLOW TO';                WT000820
                SRS_LIN(CNT)=CRD_CTR;                                   WT000830
                CNT=CNT+1;                                              WT000840
                DO WHILE((US_NTRY(UST_IDX)¬='TRM')|                     WT000850
                     (UST_PTR(UST_IDX)¬=23));                           WT000860
                     UST_IDX=UST_IDX+1;                                 WT000870
                END;                                                    WT000880
                UST_IDX=UST_IDX+1;                                      WT000890
              CRD_CTR=CRD_CTR+1;                                        WT000900
                RETURN;                                                 WT000910
            END;                                                        WT000920
                                                                        WT000930
    /* WRITE HEX ADDRESS INTO MATRIX */                                 WT000940
                                                                        WT000950
        OP2='LIT';                                                      WT000960
        AREA = UST_PTR(UST_IDX);                                        WT000970
        AREA = TRANSLATE (AREA, ZERO, BLANK);                           WT000980
        OP2_PTR = SUBSTR (AREA, 4, 4);                                  WT000990
        WRITE FILE (MATRIX) FROM (MTX_FIL);                             WT001000
                                                                        WT001010
    /* IS REMAINDER OF SOURCE LINE BLANK ? */                           WT001020
                                                                        WT001030
        UST_IDX=UST_IDX+1;                                              WT001040
        IF (US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23) THEN        WT001050
            DO;                                                         WT001060
                ERROR(CNT)='MULTIPLE ENTRIES FOR HEX ADDRESS';          WT001070
                SRS_LIN(CNT)=CRD_CTR;                                   WT001080
                CNT=CNT+1;                                              WT001090
                DO WHILE ((US_NTRY(UST_IDX)¬='TRM')|                    WT001100
```

FILE: WTOUTAR PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
                        (UST_PTR(UST_IDX)¬=23)):                    WT001110
                        UST_IDX=UST_IDX+1:                          WT001120
                    END:                                            WT001130
                    UST_IDX=UST_IDX+1:                              WT001140
                    CRD_CTR=CRD_CTR+1:                              WT001150
                    RETURN:                                         WT001160
                END:                                                WT001170
                                                                    WT001180
    /*NORMAL RETURN TO CONTROL*/                                    WT001190
                                                                    WT001200
        UST_IDX=UST_IDX+1:                                          WT001210
        CRD_CTR=CRD_CTR+1:                                          WT001220
        RETURN:                                                     WT001230
END WTOUTAR:                                                        WT001240
```

```
FILE: GOTOAR   PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

GOTOAR : PROCEDURE;                                                          GOT00010
    DCL  1   UST                    (999)     EXT,                           GOT00020
             2   US_NTRY            CHAR      (3),                           GOT00030
             2   UST_PTR            FIXED     (4);                           GOT00040
    DCL  1   ERR_TBL                (75)      EXT,                           GOT00050
             2   ERROR              CHAR      (40),                          GOT00060
             2   SRS_LIN            FIXED     (3);                           GOT00070
    DCL  1   LIT_TBL                (500)     EXT,                           GOT00080
             2   LT_NTRY            CHAR      (9);                           GOT00090
    DCL  1   MTX_FIL,                                                        GOT00100
             2   LBL_PTR            CHAR      (4),                           GOT00110
             2   OPR                CHAR      (3),                           GOT00120
             2   OPR_PTR            CHAR      (4),                           GOT00130
             2   OP1                CHAR      (3),                           GOT00140
             2   OP1_PTR            CHAR      (4),                           GOT00150
             2   OP2                CHAR      (3),                           GOT00160
             2   OP2_PTR            CHAR      (4),                           GOT00170
             2   FILL               CHAR      (55) INIT((55)' ');            GOT00180
    DCL  UST_IDX                    FIXED     (4)  EXT,                      GOT00190
         CNT                        FIXED     (2)  EXT,                      GOT00200
         CRD_CTR                    FIXED     (3)  EXT,                      GOT00210
         TGT_PTR                    FIXED     (4)  EXT,                      GOT00220
         AREA                       CHAR      (7),                           GOT00230
         ZERO0                      CHAR      (1)  INIT('0'),                GOT00240
         BLANK                      CHAR      (1)  INIT(' ');                GOT00250
    DCL  MATRIX                     FILE      RECORD;                        GOT00260
                                                                             GOT00270
    /* IS NEXT UNIFORM SYMBOL A LITERAL OF < 6 CHARACTERS ? */               GOT00280
                                                                             GOT00290
       UST_IDX = UST_IDX + 1;                                                GOT00300
       IF(US_NTRY(UST_IDX)¬='LIT')|                                          GOT00310
          (VERIFY(LT_NTRY(UST_PTR(UST_IDX)),'0123456789')>6)                 GOT00320
          THEN DO;                                                           GOT00330
               ERROR(CNT)='LINE NUMBER MUST FOLLOW GOTO STMT';               GOT00340
               SRS_LIN(CNT)=CRD_CTR;                                         GOT00350
               CNT = CNT + 1;                                                GOT00360
               DO WHILE((US_NTRY(UST_IDX)¬='TRM')|                           GOT00370
                    (UST_PTR(UST_IDX)¬=23));                                 GOT00380
                    UST_IDX = UST_IDX + 1;                                   GOT00390
               END;                                                          GOT00400
               UST_IDX = UST_IDX + 1;                                        GOT00410
               CRD_CTR = CRD_CTR + 1;                                        GOT00420
               RETURN;                                                       GOT00430
          END;                                                               GOT00440
                                                                             GOT00450
    /* CHECK FOR ILLEGAL RADIX OR ALPHABETIC */                              GOT00460
                                                                             GOT00470
       IF(VERIFY(LT_NTRY(UST_PTR(UST_IDX)),' 0123456789')¬=0) THEN           GOT00480
          DO;                                                                GOT00490
               ERROR(CNT)='LINE NUMBER MUST FOLLOW GOTO STMT';               GOT00500
               SRS_LIN(CNT) = CRD_CTR;                                       GOT00510
               CNT = CNT + 1;                                                GOT00520
               DO WHILE((US_NTRY(UST_IDX)¬='TRM')|                           GOT00530
                    (UST_PTR(UST_IDX)¬=23));                                 GOT00540
                    UST_IDX = UST_IDX + 1;                                   GOT00550
```

FILE: GOTOAR PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
                    END;                                                    GOT00560
                    UST_IDX = UST_IDX + 1;                                  GOT00570
                    CRD_CTR = CRD_CTR + 1;                                  GOT00580
                    RETURN;                                                 GOT00590
                END;                                                        GOT00600
                                                                            GOT00610
    /* IS REMAINDER OF SOURCE LINE BLANK ? */                               GOT00620
                                                                            GOT00630
        UST_IDX = UST_IDX + 1;                                              GOT00640
        IF (US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23) THEN            GOT00650
            DO;                                                             GOT00660
                ERROR(CNT)='MULTIPLE ARGUMENTS IN GOTO STMT';               GOT00670
                SRS_LIN(CNT)=CRD_CTR;                                       GOT00680
                CNT = CNT + 1;                                              GOT00690
                DO WHILE((US_NTRY(UST_IDX)¬='TRM')|                         GOT00700
                    (UST_PTR(UST_IDX)¬=23));                                GOT00710
                    UST_IDX = UST_IDX + 1;                                  GOT00720
                END;                                                        GOT00730
                UST_IDX = UST_IDX + 1;                                      GOT00740
                CRD_CTR = CRD_CTR + 1;                                      GOT00750
                RETURN;                                                     GOT00760
            END;                                                            GOT00770
                                                                            GOT00780
    /* NORMAL RETURN TO CONTROL */                                          GOT00790
                                                                            GOT00800
        AREA = TGT_PTR;                                                     GOT00810
        AREA = TRANSLATE (AREA, ZERO0, BLANK);                              GOT00820
        LBL_PTR = SUBSTR (AREA, 4, 4);                                      GOT00830
        OPR = 'KEY';                                                        GOT00840
        OPR_PTR = '0002';                                                   GOT00850
        OP1 = 'LIT';                                                        GOT00860
        AREA = UST_PTR(UST_IDX-1);                                          GOT00870
        AREA = TRANSLATE (AREA, ZERO0, BLANK);                              GOT00880
        OP1_PTR = SUBSTR (AREA, 4, 4);                                      GOT00890
        OP2 = '    ';                                                       GOT00900
        OP2_PTR = '0000';                                                   GOT00910
        WRITE FILE (MATRIX) FROM (MTX_FIL);                                 GOT00920
        UST_IDX = UST_IDX + 1;                                              GOT00930
        CRD_CTR = CRD_CTR + 1;                                              GOT00940
        RETURN;                                                             GOT00950
END GOTOAR;                                                                 GOT00960
```

FILE: RESTRAR PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
RESTRAR : PROCEDURE:                                                    RES00010
   DCL  1   UST                          (999)    EXT,                  RES00020
            2   US_NTRY                  CHAR     (3),                  RES00030
            2   UST_PTR                  FIXED    (4);                  RES00040
   DCL  1   ERR_TBL                      (75)     EXT,                  RES00050
            2   ERROR                    CHAR     (40),                 RES00060
            2   SRS_LIN                  FIXED    (3);                  RES00070
   DCL  1   MTX_FIL,                                                    RES00080
            2   LBL_PTR                  CHAR     (4),                  RES00090
            2   OPR                      CHAR     (3),                  RES00100
            2   OPR_PTR                  CHAR     (4),                  RES00110
            2   OP1                      CHAR     (3),                  RES00120
            2   OP1_PTR                  CHAR     (4),                  RES00130
            2   OP2                      CHAR     (3),                  RES00140
            2   OP2_PTR                  CHAR     (4),                  RES00150
            2   FILL                     CHAR     (55) INIT((55)' '):   RES00160
   DCL  UST_IDX                          FIXED    (4)  EXT,             RES00170
        CNT                              FIXED    (2)  EXT,             RES00180
        AREA                             CHAR     (7),                  RES00190
        ZERO0                            CHAR     (1)  INIT('0'),       RES00200
        BLANK                            CHAR     (1)  INIT(' '),       RES00210
        CRD_CTR                          FIXED    (3)  EXT,             RES00220
        TGT_PTR                          FIXED    (4)  EXT;             RES00230
   DCL  MATRIX                           FILE     RECORD;               RES00240
                                                                         RES00250
   /* THE NEXT UNIFORM SYMBOL MUST BE '$' */                            RES00260
                                                                         RES00270
        UST_IDX = UST_IDX + 1;                                           RES00280
        IF(US_NTRY(UST_IDX) ¬= 'TRM')|(UST_PTR(UST_IDX)¬=23) THEN        RES00290
             GO TO RS001;                                                RES00300
        AREA = TGT_PTR;                                                  RES00310
        AREA = TRANSLATE (AREA, ZERO0, BLANK);                           RES00320
        LBL_PTR = SUBSTR (AREA, 4, 4);                                   RES00330
        OPR = 'KEY';                                                     RES00340
        OPR_PTR = '0014';                                                RES00350
        OP1 = '   ';                                                     RES00360
        OP1_PTR = '0000';                                                RES00370
        OP2 = '   ';                                                     RES00380
        OP2_PTR = '0000';                                                RES00390
        WRITE FILE (MATRIX) FROM (MTX_FIL);                              RES00400
        UST_IDX = UST_IDX + 1;                                           RES00410
        CRD_CTR = CRD_CTR + 1;                                           RES00420
        RETURN;                                                          RES00430
 RS001:ERROR(CNT) = 'CHARACTERS APPEAR AFTER RESTORE STMT';              RES00440
        SRS_LIN(CNT) = CRD_CTR;                                          RES00450
        CNT = CNT + 1;                                                   RES00460
        DO WHILE((US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23));      RES00470
             UST_IDX = UST_IDX + 1;                                      RES00480
        END;                                                             RES00490
        UST_IDX = UST_IDX + 1;                                           RES00500
        CRD_CTR = CRD_CTR + 1;                                           RES00510
        RETURN;                                                          RES00520
END RESTRAR;                                                             RES00530
```

FILE: READAR PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
READAR : PROCEDURE;                                                    READ00010
   DCL  1   UST                          (999)   EXT,                  READ00020
            2   US_NTRY                  CHAR    (3),                  READ00030
            2   UST_PTR                  FIXED   (4);                  READ00040
   DCL  1   ERR_TBL                      (75)    EXT,                  READ00050
            2   ERROR                    CHAR    (40),                 READ00060
            2   SRS_LIN                  FIXED   (3);                  READ00070
   DCL  1   MTX_FIL,                                                    READ00080
            2   LBL_PTR                  CHAR    (4),                  READ00090
            2   OPR                      CHAR    (3),                  READ00100
            2   OPR_PTR                  CHAR    (4),                  READ00110
            2   OP1                      CHAR    (3),                  READ00120
            2   OP1_PTR                  CHAR    (4),                  READ00130
            2   OP2                      CHAR    (3),                  READ00140
            2   OP2_PTR                  CHAR    (4),                  READ00150
            2   FILL                     CHAR    (55) INIT((55)' '):   READ00160
   DCL  UST_IDX                          FIXED   (4)  EXT,             READ00170
        CNT                              FIXED   (2)  EXT,             READ00180
        AREA                             CHAR    (7),                  READ00190
        ZERO0                            CHAR    (1)  INIT('0'),       READ00200
        BLANK                            CHAR    (1)  INIT(' '),       READ00210
        CRD_CTR                          FIXED   (3)  EXT,             READ00220
        TGT_PTR                          FIXED   (4)  EXT;             READ00230
   DCL  MATRIX                           FILE    RECORD;               READ00240
                                                                       READ00250
   /* IS THE NEXT UNIFORM SYMBOL AN IDENTIFIER ? */                     READ00260
                                                                       READ00270
        UST_IDX = UST_IDX + 1;                                         READ00280
   RD001:IF US_NTRY(UST_IDX)¬='IDN' THEN GO TO RD002;                  READ00290
                                                                       READ00300
   /* ENTER IDENTIFIER INTO MATRIX */                                  READ00310
                                                                       READ00320
        AREA = TGT_PTR;                                                READ00330
        AREA = TRANSLATE (AREA, ZERO0, BLANK);                         READ00340
        LBL_PTR = SUBSTR (AREA, 4, 4);                                 READ00350
        OPR='KEY';                                                     READ00360
        OPR_PTR='0013';                                                READ00370
        OP1='IDN';                                                     READ00380
        AREA = UST_PTR(UST_IDX);                                       READ00390
        AREA = TRANSLATE (AREA, ZERO0, BLANK);                         READ00400
        OP1_PTR = SUBSTR (AREA, 4, 4);                                 READ00410
        OP2='   ';                                                    READ00420
        OP2_PTR='0000';                                                READ00430
        WRITE FILE (MATRIX) FROM (MTX_FIL);                            READ00440
        UST_IDX=UST_IDX+1;                                             READ00450
                                                                       READ00460
   /* IS NEW UNIFORM SYMBOL A COMMA ? */                               READ00470
                                                                       READ00480
        IF (US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=11)THEN        READ00490
            GO TO RD003;                                               READ00500
        UST_IDX=UST_IDX+1;                                             READ00510
        GO TO RD001;                                                   READ00520
   RD002:ERROR(CNT) = 'READ ARGUMENT MUST BE A VARIABLE NAME';         READ00530
        SRS_LIN(CNT)=CRD_CTR;                                          READ00540
        CNT=CNT+1;                                                     READ00550
```

```
FILE: READAR   PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

        DO WHILE((US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23));    REA00560
            UST_IDX=UST_IDX + 1;                                       REA00570
        END;                                                           REA00580
        UST_IDX=UST_IDX+1;                                             REA00590
        CRD_CTR=CRD_CTR+1;                                             REA00600
        RETURN;                                                        REA00610
    RD003:IF (US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23) THEN     REA00620
        DO:  ERROR(CNT)='COMMAS REQUIRED BETWEEN READ ARGUMENTS';      REA00630
            SRS_LIN(CNT)=CRD_CTR;                                      REA00640
            CNT=CNT+1;                                                 REA00650
            DO WHILE((US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=     REA00660
                23));                                                  REA00670
                UST_IDX=UST_IDX+1;                                     REA00680
            END;                                                       REA00690
            UST_IDX=UST_IDX+1;                                         REA00700
            CRD_CTR=CRD_CTR+1;                                         REA00710
            RETURN;                                                    REA00720
        END;                                                           REA00730
                                                                       REA00740
    /* NORMAL RETURN TO CONTROL */                                     REA00750
                                                                       REA00760
        UST_IDX = UST_IDX + 1;                                         REA00770
        CRD_CTR = CRD_CTR + 1;                                         REA00780
        RETURN;                                                        REA00790
END READAR;                                                            REA00800
```

```
FILE: ENDAR     PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

ENDAR : PROCEDURE;                                                      END00010
   DCL  1    UST                          (999)     EXT,              END00020
             2    US_NTRY                 CHAR      (3),              END00030
             2    UST_PTR                 FIXED     (4);              END00040
   DCL  1    ERR_TBL                      (75)      EXT,              END00050
             2    ERROR                   CHAR      (40),             END00060
             2    SRS_LIN                 FIXED     (3);              END00070
   DCL  1    MTX_FIL,                                                 END00080
             2    LBL_PTR                 CHAR      (4),              END00090
             2    OPR                     CHAR      (3),              END00100
             2    OPR_PTR                 CHAR      (4),              END00110
             2    OP1                     CHAR      (3),              END00120
             2    OP1_PTR                 CHAR      (4),              END00130
             2    OP2                     CHAR      (3),              END00140
             2    OP2_PTR                 CHAR      (4),              END00150
             2    FILL                    CHAR      (55) INIT((55)' '); END00160
   DCL  UST_IDX                           FIXED     (4)  EXT,         END00170
        CNT                               FIXED     (2)  EXT,         END00180
        AREA                              CHAR      (7),              END00190
        ZERO0                             CHAR      (1)  INIT('0'),   END00200
        BLANK                             CHAR      (1)  INIT(' '),   END00210
        CRD_CTR                           FIXED     (3)  EXT,         END00220
        TGT_PTR                           FIXED     (4)  EXT;         END00230
   DCL  MATRIX                            FILE      RECORD;           END00240
                                                                       END00250
    /* THE NEXT UNIFORM SYMBOL MUST BE '$' */                          END00260
                                                                       END00270
        UST_IDX = UST_IDX + 1;                                         END00280
        IF(US_NTRY(UST_IDX) ¬= 'TRM')|(UST_PTR(UST_IDX)¬=23) THEN      END00290
             GO TO EA001;                                              END00300
        AREA = TGT_PTR;                                                END00310
        AREA = TRANSLATE (AREA, ZERO0, BLANK);                         END00320
        LBL_PTR = SUBSTR (AREA, 4, 4);                                 END00330
        OPR = 'KEY';                                                   END00340
        OPR_PTR = '0017';                                              END00350
        OP1 = ' ';                                                     END00360
        OP1_PTR = '0000';                                              END00370
        OP2 = ' ';                                                     END00380
        OP2_PTR = '0000';                                              END00390
        WRITE FILE (MATRIX) FROM (MTX_FIL);                            END00400
        UST_IDX = UST_IDX + 1;                                         END00410
        CRD_CTR = CRD_CTR + 1;                                         END00420
        RETURN;                                                        END00430
   EA001:ERROR(CNT) = 'CHARACTERS APPEAR AFTER END STMT';              END00440
        SRS_LIN(CNT) = CRD_CTR;                                        END00450
        CNT = CNT + 1;                                                 END00460
        DO WHILE((US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23));    END00470
             UST_IDX = UST_IDX + 1;                                    END00480
        END;                                                           END00490
        UST_IDX = UST_IDX + 1;                                         END00500
        CRD_CTR = CRD_CTR + 1;                                         END00510
        RETURN;                                                        END00520
 END ENDAR;                                                            END00530
```

```
FILE: FORAR     PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

FORAR : PROCEDURE;                                                          FOR00010
    DCL  1    ERR_TBL                (75)      EXT,                       FOR00020
              2    ERROR               CHAR      (40),                    FOR00030
              2    SRS_LIN             FIXED     (3);                     FOR00040
    DCL  1    UST                    (999)      EXT,                       FOR00050
              2    US_NTRY             CHAR      (3),                     FOR00060
              2    UST_PTR             FIXED     (4);                     FOR00070
    DCL  1    LIT_TBL                (500)      EXT,                       FOR00080
              2    LT_NTRY             CHAR      (9);                     FOR00090
    DCL  1    MTX_FIL,                                                    FOR00100
              2    LBL_PTR             CHAR      (4),                     FOR00110
              2    OPR                 CHAR      (3),                     FOR00120
              2    OPR_PTR             CHAR      (4),                     FOR00130
              2    OP1                 CHAR      (3),                     FOR00140
              2    OP1_PTR             CHAR      (4),                     FOR00150
              2    OP2                 CHAR      (3),                     FOR00160
              2    OP2_PTR             CHAR      (4),                     FOR00170
              2    FILL                CHAR      (55)  INIT((55)' '):     FOR00180
    DCL  1    FOR_STK                CONTROLLED        EXT,               FOR00190
              2    FOR_LINE_PTR        FIXED     (4),                     FOR00200
              2    FOR_VARIABLE_PTR    FIXED     (4),                     FOR00210
              2    STEP_PTR            FIXED     (4);                     FOR00220
    DCL  UST_IDX                      FIXED     (4)   EXT,                FOR00230
         CNT                          FIXED     (2)   EXT,                FOR00240
         AREA                         CHAR      (7),                      FOR00250
         ZERO0                        CHAR      (1)   INIT('0'),          FOR00260
         BLANK                        CHAR      (1)   INIT(' '),          FOR00270
         CRD_CTR                      FIXED     (3)   EXT,                FOR00280
         TGT_PTR                      FIXED     (4)   EXT,                FOR00290
         VAR_NAME                     FIXED     (4),                      FOR00300
         RETURN_IDX                   FIXED     (4),                      FOR00310
         LOOP_EXIT                    FIXED     (4),                      FOR00320
         STEP                         FIXED     (4),                      FOR00330
         N                            FIXED     (2);                      FOR00340
    DCL  MATRIX                       FILE      RECORD;                   FOR00350
                                                                          FOR00360
    /* NEXT SYMBOL MUST BE AN IDENTIFIER */                               FOR00370
                                                                          FOR00380
         UST_IDX = UST_IDX + 1;                                           FOR00390
         IF   US_NTRY(UST_IDX)¬='IDN' THEN                                FOR00400
              DO;  ERROR(CNT)='IDENTIFIER MUST FOLLOW FOR';               FOR00410
                   GO TO FA001_ERROR;                                     FOR00420
              END;                                                        FOR00430
         VAR_NAME = UST_PTR(UST_IDX);                                     FOR00440
                                                                          FOR00450
    /* NEXT SYMBOL MUST BE '=' */                                         FOR00460
                                                                          FOR00470
         UST_IDX = UST_IDX + 1;                                           FOR00480
         IF(US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=6) THEN           FOR00490
              DO;  ERROR(CNT)='EQUAL SIGN IS NOT IN PROPER POSITION';     FOR00500
                   GO TO FA001_ERROR;                                     FOR00510
              END;                                                        FOR00520
                                                                          FOR00530
    /* NEXT SYMBOL MUST BE A POSITIVE INTEGER OR VARIABLE */              FOR00540
                                                                          FOR00550
```

```
FILE: FORAR     PLIOPT    A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

        UST_IDX = UST_IDX + 1;                                                  FOR00560
        IF(US_NTRY(UST_IDX)¬='IDN')&(US_NTRY(UST_IDX)¬='LIT') THEN              FOR00570
            DO:  ERROR(CNT)='POS.INTEGER OR VAR. MUST PRECEDE TO':              FOR00580
                 GO TO FA001_ERROR:                                             FOR00590
            END:                                                                FOR00600
        IF(US_NTRY(UST_IDX)='LIT')&                                             FOR00610
          (VERIFY(LT_NTRY(UST_PTR(UST_IDX)),' 0123456789')¬=0) THEN             FOR00620
            DO:  ERROR(CNT)='LITERAL PRECEDING TO IS NOT INTEGER':              FOR00630
                 GO TO FA001_ERROR:                                             FOR00640
            END:                                                                FOR00650
                                                                                FOR00660
    /* WRITE INTO MATRIX SETTING VARIABLE EQUAL TO TOKEN */                     FOR00670
                                                                                FOR00680
        AREA = TGT_PTR:                                                         FOR00690
        AREA = TRANSLATE(AREA,ZERO0,BLANK);                                     FOR00700
        LAL_PTR = SUBSTR(AREA, 4, 4);                                           FOR00710
        OPR = 'TRM';                                                            FOR00720
        OPR_PTR = '0005';                                                       FOR00730
        OP1 = 'IDN';                                                            FOR00740
        AREA = VAR_NAME;                                                        FOR00750
        AREA = TRANSLATE(AREA,ZERO0,BLANK);                                     FOR00760
        OP1_PTR = SUBSTR (AREA, 4, 4);                                          FOR00770
        OP2 = US_NTRY(UST_IDX);                                                 FOR00780
        AREA = UST_PTR(UST_IDX);                                                FOR00790
        AREA = TRANSLATE(AREA,ZERO0,BLANK);                                     FOR00800
        OP2_PTR = SUBSTR (AREA, 4, 4);                                          FOR00810
        WRITE FILE (MATRIX) FROM (MTX_FIL);                                     FOR00820
                                                                                FOR00830
    /* NEXT SYMBOL MUST BE 'TO' */                                              FOR00840
                                                                                FOR00850
        UST_IDX = UST_IDX + 1;                                                  FOR00860
        IF(US_NTRY(UST_IDX)¬='KEY')|(UST_PTR(UST_IDX)¬=4) THEN                  FOR00870
            DO:  ERROR(CNT)='TO IS MISPLACED OR MISSING':                       FOR00880
                 GO TO FA001_ERROR:                                             FOR00890
            END:                                                                FOR00900
                                                                                FOR00910
    /* DETERMINE NUMBER OF LINE FOLLOWING ASSOCIATED 'NEXT' */                  FOR00920
                                                                                FOR00930
        RETURN_IDX=UST_IDX;                                                     FOR00940
        DO WHILE (US_NTRY(UST_IDX)¬=' '):                                       FOR00950
FA002:      UST_IDX=UST_IDX+1;                                                  FOR00960
            IF(US_NTRY(UST_IDX)='KEY')&(UST_PTR(UST_IDX)=7)&                    FOR00970
                (US_NTRY(UST_IDX+1)='IDN')&(UST_PTR(UST_IDX+1)=                 FOR00980
                VAR_NAME) THEN                                                  FOR00990
                    DO:                                                         FOR01000
                        LOOP_EXIT=UST_PTR(UST_IDX+3);                           FOR01010
                        UST_IDX = RETURN_IDX;                                   FOR01020
                        GO TO FA003:                                            FOR01030
                    END;                                                        FOR01040
            GO TO FA002:                                                        FOR01050
        END;                                                                    FOR01060
        ERROR(CNT) = 'MISSING NEXT STATEMENT':                                  FOR01070
        GO TO FA001_ERROR:                                                      FOR01080
FA003:                                                                          FOR01090
                                                                                FOR01100
```

```
FILE: FORAR     PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

     /* NEXT SYMBOL MUST BE A POSITIVE INTEGER OR VARIABLE */                FOR01110
          UST_IDX = UST_IDX + 1;                                             FOR01120
          IF(US_NTRY(UST_IDX)¬='IDN')&(US_NTRY(UST_IDX)¬='LIT') THEN         FOR01130
               DO:  ERROR(CNT)='POS. INTEGER OR VAR. MUST FOLLOW TO':        FOR01140
                    GO TO FA001_ERROR:                                       FOR01150
               END:                                                          FOR01160
          IF(US_NTRY(UST_IDX)='LIT')&(VERIFY(LT_NTRY(UST_PTR(UST_IDX)),      FOR01170
               ' 0123456789')¬=0) THEN                                       FOR01180
               DO:  ERROR(CNT)='LITERAL FOLLOWING TO IS NOT INTEGER':        FOR01190
                    GO TO FA001_ERROR:                                       FOR01200
               END:                                                          FOR01210
                                                                             FOR01220
     /* WRITE MATRIX LINE TO COMPARE VALUE OF VARIABLE TO LIMIT */           FOR01230
                                                                             FOR01240
          OPR_PTR='0024':                                                    FOR01250
          OP2=US_NTRY(UST_IDX):                                              FOR01260
          AREA = UST_PTR(UST_IDX):                                           FOR01270
          AREA = TRANSLATE(AREA, ZERO0, BLANK):                              FOR01280
          OP2_PTR = SUBSTR(AREA, 4, 4):                                      FOR01290
          WRITE FILE (MATRIX) FROM (MTX_FIL):                                FOR01300
                                                                             FOR01310
     /* WRITE MATRIX LINE TO EXIT LOOP */                                    FOR01320
                                                                             FOR01330
          OPR = 'KEY':                                                       FOR01340
          OPR_PTR = '0002':                                                  FOR01350
          OP1 = 'LIT':                                                       FOR01360
          AREA = LOOP_EXIT:                                                  FOR01370
          AREA = TRANSLATE(AREA, ZERO0, BLANK):                              FOR01380
          OP1_PTR = SUBSTR(AREA, 4, 4):                                      FOR01390
          OP2 = '    ':                                                      FOR01400
          OP2_PTR = '0000':                                                  FOR01410
          WRITE FILE (MATRIX) FROM (MTX_FIL):                                FOR01420
                                                                             FOR01430
     /* IS NEXT SYMBOL '$' ? */                                              FOR01440
                                                                             FOR01450
          UST_IDX = UST_IDX + 1:                                             FOR01460
          IF(US_NTRY(UST_IDX)='TRM')&(UST_PTR(UST_IDX)=23) THEN              FOR01470
               DO:  STEP = 1:                                                FOR01480
                    GO TO FA004:                                             FOR01490
               END:                                                          FOR01500
                                                                             FOR01510
     /* IS SYMBOL 'STEP' ? */                                                FOR01520
                                                                             FOR01530
          IF(US_NTRY(UST_IDX)¬='KEY')|(UST_PTR(UST_IDX)¬=10) THEN            FOR01540
               DO:  ERROR(CNT)='ONLY STEP MAY FOLLOW INTEGER AFTER TO':      FOR01550
                    GO TO FA001_ERROR:                                       FOR01560
               END:                                                          FOR01570
                                                                             FOR01580
     /* IS NEXT SYMBOL A POSITIVE INTEGER OR VARIABLE ? */                   FOR01590
                                                                             FOR01600
          UST_IDX = UST_IDX + 1:                                             FOR01610
          IF(US_NTRY(UST_IDX)¬='IDN')&(US_NTRY(UST_IDX)¬='LIT') THEN         FOR01620
               DO:  ERROR(CNT)='POSITIVE INTEGER MUST FOLLOW STEP':          FOR01630
                    GO TO FA001_ERROR:                                       FOR01640
               END:                                                          FOR01650
```

```
FILE: FORAR     PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

      IF(US_NTRY(UST_IDX)='LIT')&                                          FOR01660
        (VERIFY(LT_NTRY(UST_PTR(UST_IDX)),' 0123456789')¬=0)THEN           FOR01670
          DO:  ERROR(CNT)='LITERAL FOLLOWING STEP IS NOT INTEGER';         FOR01680
               GO TO FA001_ERROR;                                          FOR01690
          END;                                                             FOR01700
                                                                           FOR01710
   /* IS NEXT SYMBOL '$' ? */                                             FOR01720
                                                                           FOR01730
      UST_IDX = UST_IDX + 1;                                               FOR01740
      IF(US_NTRY(UST_IDX)¬='TRM')&(UST_PTR(UST_IDX)¬=23) THEN              FOR01750
          DO:  ERROR(CNT)='EXTRANEOUS CHARACTERS AT END OF LINE';          FOR01760
               GO TO FA001_ERROR;                                          FOR01770
          END;                                                             FOR01780
                                                                           FOR01790
      STEP = UST_PTR(UST_IDX-1);                                           FOR01800
                                                                           FOR01810
   FA004:                                                                  FOR01820
                                                                           FOR01830
   /* PUSH LINE NUMBER, VARIABLE NAME, AND STEP VALUE ONTO STACK */        FOR01840
                                                                           FOR01850
      ALLOCATE FOR_STK;                                                    FOR01860
      FOR_LINE_PTR = TGT_PTR;                                              FOR01870
      FOR_VARIABLE_PTR = VAR_NAME;                                         FOR01880
      STEP_PTR = STEP;                                                     FOR01890
                                                                           FOR01900
   /* NORMAL RETURN TO CONTROL */                                          FOR01910
                                                                           FOR01920
      UST_IDX = UST_IDX + 1;                                               FOR01930
      CRD_CTR = CRD_CTR + 1;                                               FOR01940
      RETURN;                                                              FOR01950
                                                                           FOR01960
   /* ERROR ROUTINE */                                                     FOR01970
                                                                           FOR01980
   FA001_ERROR:                                                            FOR01990
      SRS_LIN(CNT)=CRD_CTR;                                                FOR02000
      CNT = CNT + 1;                                                       FOR02010
      DO WHILE ((US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23));         FOR02020
          UST_IDX = UST_IDX + 1;                                           FOR02030
      END;                                                                 FOR02040
      UST_IDX = UST_IDX + 1;                                               FOR02050
      CRD_CTR = CRD_CTR + 1;                                               FOR02060
      RETURN;                                                              FOR02070
   END FORAR;                                                              FOR02080
```

FILE: NEXTAR PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
NEXTAR : PROCEDURE:                                                    NEX00010
    DCL  1    UST                          (999)     EXT,              NEX00020
               2    US_NTRY                CHAR      (3),              NEX00030
               2    UST_PTR                FIXED     (4):              NEX00040
    DCL  1    ERR_TBL                      (75)      EXT,              NEX00050
               2    ERROR                  CHAR      (40),             NEX00060
               2    SRS_LIN                FIXED     (3):              NEX00070
    DCL  1    MTX_FIL,                                                 NEX00080
               2    LBL_PTR                CHAR      (4),              NEX00090
               2    OPR                    CHAR      (3),              NEX00100
               2    OPR_PTR                CHAR      (4),              NEX00110
               2    OP1                    CHAR      (3),              NEX00120
               2    OP1_PTR                CHAR      (4),              NEX00130
               2    OP2                    CHAR      (3),              NEX00140
               2    OP2_PTR                CHAR      (4),              NEX00150
               2    FILL                   CHAR      (55) INIT((55)' '): NEX00160
    DCL  UST_IDX                           FIXED     (4)  EXT,         NEX00170
         CNT                               FIXED     (2)  EXT,         NEX00180
         CRD_CTR                           FIXED     (3)  EXT,         NEX00190
         TGT_PTR                           FIXED     (4)  EXT,         NEX00200
         TMP_STR                           FIXED     (3)  EXT,         NEX00210
         AREA                              CHAR      (7),              NEX00220
         ZERO0                             CHAR      (1)  INIT('0'),   NEX00230
         BLANK                             CHAR      (1)  INIT(' '):   NEX00240
    DCL  MATRIX                            FILE      RECORD:           NEX00250
    DCL  1    FOR_STK                      CONTROLLED      EXT,        NEX00260
               2    FOR_LINE_PTR           FIXED     (4),              NEX00270
               2    FOR_VARIABLE_PTR       FIXED     (4),              NEX00280
               2    STEP_PTR               FIXED     (4):              NEX00290
                                                                       NEX00300
/* IS NEXT UNIFORM SYMBOL AN IDENTIFIER ? */                           NEX00310
                                                                       NEX00320
     UST_IDX = UST_IDX + 1:                                            NEX00330
     IF US_NTRY(UST_IDX) ¬= 'IDN' THEN                                 NEX00340
          DO:                                                          NEX00350
               ERROR(CNT)='NEXT MUST BE FOLLOWED BY A VARIABLE':       NEX00360
               GO TO NA001_ERROR:                                      NEX00370
          END:                                                         NEX00380
                                                                       NEX00390
/* DOES VARIABLE MATCH TOP OF STACK ? */                              NEX00400
                                                                       NEX00410
     IF UST_PTR(UST_IDX)¬= FOR_VARIABLE_PTR THEN                       NEX00420
          DO:  ERROR(CNT) = 'IMPROPER FOR - NEXT PAIR':                NEX00430
               GO TO NA001_ERROR:                                      NEX00440
          END:                                                         NEX00450
                                                                       NEX00460
/* WRITE MATRIX LINES TO ADD STEP VALUE TO VARIABLE */                NEX00470
                                                                       NEX00480
               AREA = TGT_PTR:                                         NEX00490
               AREA = TRANSLATE(AREA, ZERO0, BLANK):                   NEX00500
               LBL_PTR = SUBSTR(AREA, 4, 4):                           NEX00510
               OPR='TRM':                                              NEX00520
               OPR_PTR='0002':                                         NEX00530
               OP1='IDN':                                              NEX00540
               AREA = UST_PTR(UST_IDX):                                NEX00550
```

```
FILE: NEXTAR   PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

                    AREA = TRANSLATE(AREA, ZERO0, BLANK);                    NEX00560
                    OP1_PTR = SUBSTR(AREA, 4, 4);                            NEX00570
                    OP2 = 'LIT';                                             NEX00580
                    AREA = STEP_PTR;                                         NEX00590
                    AREA = TRANSLATE(AREA, ZERO0, BLANK);                    NEX00600
                    OP2_PTR = SUBSTR(AREA, 4, 4);                            NEX00610
                    WRITE FILE (MATRIX) FROM (MTX_FIL);                      NEX00620
                                                                             NEX00630
                    OPR_PTR = '0006';                                        NEX00640
                    OP2 = 'TMP';                                             NEX00650
                    OP2_PTR = '0001';                                        NEX00660
                    WRITE FILE (MATRIX) FROM (MTX_FIL);                      NEX00670
                                                                             NEX00680
     /* WRITE MATRIX LINE TO RETURN EXECUTION TO TOP OF LOOP */              NEX00690
                                                                             NEX00700
          OPR = 'KEY';                                                       NEX00710
          OPR_PTR = '0002';                                                  NEX00720
          OP1 = 'LIT';                                                       NEX00730
          AREA = FOR_LINE_PTR;                                               NEX00740
          AREA = TRANSLATE(AREA, ZERO0, BLANK);                              NEX00750
          OP1_PTR = SUBSTR(AREA, 4, 4);                                      NEX00760
          OP2 = 'LIT';                                                       NEX00770
          OP2_PTR = '0002';                                                  NEX00780
          WRITE FILE (MATRIX) FROM (MTX_FIL);                                NEX00790
                                                                             NEX00800
     /* POP UP NEXT 'FOR' INFORMATION */                                     NEX00810
                                                                             NEX00820
          FREE FOR_STK;                                                      NEX00830
                                                                             NEX00840
     /* IS NEXT SYMBOL '$' ? */                                             NEX00850
                                                                             NEX00860
          UST_IDX = UST_IDX + 1;                                             NEX00870
          IF(US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23) THEN            NEX00880
               DO;  ERROR(CNT)='CHARACTERS APPEAR AFTER VARIABLE NAME';      NEX00890
                    GO TO NA001_ERROR;                                       NEX00900
               END;                                                          NEX00910
                                                                             NEX00920
     /* NORMAL RETURN TO CONTROL */                                          NEX00930
                                                                             NEX00940
          UST_IDX = UST_IDX + 1;                                             NEX00950
          CRD_CTR = CRD_CTR + 1;                                             NEX00960
          RETURN;                                                            NEX00970
                                                                             NEX00980
     /* ERROR ROUTINE */                                                    NEX00990
                                                                             NEX01000
     NA001_ERROR:                                                            NEX01010
          SRS_LIN(CNT) = CRD_CTR;                                            NEX01020
          CNT = CNT + 1;                                                     NEX01030
          DO WHILE ((US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23));       NEX01040
               UST_IDX=UST_IDX+1;                                            NEX01050
          END;                                                               NEX01060
          UST_IDX = UST_IDX + 1;                                             NEX01070
          CRD_CTR = CRD_CTR + 1;                                             NEX01080
          RETURN;                                                            NEX01090
     END NEXTAR;                                                             NEX01100
```

```
FILE: GOSUBAR  PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

GOSUBAR : PROCEDURE:                                                    GOS00010
    DCL  1    UST                        (999)     EXT,                 GOS00020
              2    US_NTRY               CHAR      (3),                 GOS00030
              2    UST_PTR               FIXED     (4);                 GOS00040
    DCL  1    ERR_TBL                    (75)      EXT,                 GOS00050
              2    ERROR                 CHAR      (40),                GOS00060
              2    SRS_LIN               FIXED     (3);                 GOS00070
    DCL  1    LIT_TBL                    (500)     EXT,                 GOS00080
              2    LT_NTRY               CHAR      (9);                 GOS00090
    DCL  1    MTX_FIL,                                                  GOS00100
              2    LBL_PTR               CHAR      (4),                 GOS00110
              2    OPR                   CHAR      (3),                 GOS00120
              2    OPR_PTR               CHAR      (4),                 GOS00130
              2    OP1                   CHAR      (3),                 GOS00140
              2    OP1_PTR               CHAR      (4),                 GOS00150
              2    OP2                   CHAR      (3),                 GOS00160
              2    OP2_PTR               CHAR      (4),                 GOS00170
              2    FILL                  CHAR      (55) INIT((55)' '):  GOS00180
    DCL  UST_IDX                         FIXED     (4)  EXT,            GOS00190
         CNT                             FIXED     (2)  EXT,            GOS00200
         CRD_CTR                         FIXED     (3)  EXT,            GOS00210
         TGT_PTR                         FIXED     (4)  EXT,            GOS00220
         AREA                            CHAR      (7),                 GOS00230
         ZERO0                           CHAR      (1)  INIT('0'),      GOS00240
         BLANK                           CHAR      (1)  INIT(' '):      GOS00250
    DCL  MATRIX                          FILE      RECORD;              GOS00260
                                                                        GOS00270
    /* IS NEXT UNIFORM SYMBOL A LITERAL OF < 6 CHARACTERS ? */          GOS00280
                                                                        GOS00290
         UST_IDX = UST_IDX + 1;                                         GOS00300
         IF(US_NTRY(UST_IDX)¬='LIT')|                                   GOS00310
              (VERIFY(LT_NTRY(UST_PTR(UST_IDX)),'0123456789')>6)        GOS00320
              THEN DO:                                                  GOS00330
                   ERROR(CNT)='LINE NUMBER MUST FOLLOW GOSUB STMT':     GOS00340
                   SRS_LIN(CNT)=CRD_CTR;                                GOS00350
                   CNT = CNT + 1;                                       GOS00360
                   DO WHILE((US_NTRY(UST_IDX)¬='TRM')|                  GOS00370
                        (UST_PTR(UST_IDX)¬=23));                        GOS00380
                        UST_IDX = UST_IDX + 1;                          GOS00390
                   END;                                                 GOS00400
                   UST_IDX = UST_IDX + 1;                               GOS00410
                   CRD_CTR = CRD_CTR + 1;                               GOS00420
                   RETURN;                                              GOS00430
              END;                                                      GOS00440
                                                                        GOS00450
    /* CHECK FOR ILLEGAL RADIX OR ALPHABETIC */                         GOS00460
                                                                        GOS00470
         IF(VERIFY(LT_NTRY(UST_PTR(UST_IDX)),' 0123456789')¬=0) THEN    GOS00480
              DO:                                                       GOS00490
                   ERROR(CNT)='LINE NUMBER MUST FOLLOW GOSUB STMT':     GOS00500
                   SRS_LIN(CNT)=CRD_CTR;                                GOS00510
                   CNT = CNT + 1;                                       GOS00520
                   DO WHILE((US_NTRY(UST_IDX)¬='TRM')|                  GOS00530
                        (UST_PTR(UST_IDX)¬=23));                        GOS00540
                        UST_IDX = UST_IDX + 1;                          GOS00550
```

```
FILE: GOSUBAR  PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

                    END;                                                    GOS00560
                    UST_IDX = UST_IDX + 1;                                  GOS00570
                    CRD_CTR = CRD_CTR + 1;                                  GOS00580
                    RETURN;                                                 GOS00590
                END;                                                        GOS00600
                                                                            GOS00610
    /* IS REMAINDER OF SOURCE LINE BLANK ? */                               GOS00620
                                                                            GOS00630
        UST_IDX = UST_IDX + 1;                                              GOS00640
        IF (US_NTRY(UST_IDX)¬='TRM')|(UST_PTR(UST_IDX)¬=23) THEN            GOS00650
            DO;                                                             GOS00660
                ERROR(CNT)='MULTIPLE ARGUMENTS IN GOSUB STMT';              GOS00670
                SRS_LIN(CNT)=CRD_CTR;                                       GOS00680
                CNT = CNT + 1;                                              GOS00690
                DO WHILE((US_NTRY(UST_IDX)¬='TRM')|                         GOS00700
                    (UST_PTR(UST_IDX)¬=23));                                GOS00710
                    UST_IDX = UST_IDX + 1;                                  GOS00720
                END;                                                        GOS00730
                UST_IDX = UST_IDX + 1;                                      GOS00740
                CRD_CTR = CRD_CTR + 1;                                      GOS00750
                RETURN;                                                     GOS00760
            END;                                                            GOS00770
                                                                            GOS00780
    /* NORMAL RETURN TO CONTROL */                                          GOS00790
                                                                            GOS00800
        AREA = TGT_PTR;                                                     GOS00810
        AREA = TRANSLATE (AREA, ZERO0, BLANK);                              GOS00820
        LBL_PTR = SUBSTR (AREA, 4, 4);                                      GOS00830
        OPR = 'KEY';                                                        GOS00840
        OPR_PTR = '0008';                                                   GOS00850
        OP1 = 'LIT';                                                        GOS00860
        AREA = UST_PTR(UST_IDX-1);                                          GOS00870
        AREA = TRANSLATE (AREA, ZERO0, BLANK);                              GOS00880
        OP1_PTR = SUBSTR (AREA, 4, 4);                                      GOS00890
        OP2 = '    ';                                                       GOS00900
        OP2_PTR = '0000';                                                   GOS00910
        WRITE FILE (MATRIX) FROM (MTX_FIL);                                 GOS00920
        UST_IDX = UST_IDX + 1;                                              GOS00930
        CRD_CTR = CRD_CTR + 1;                                              GOS00940
        RETURN;                                                             GOS00950
END GOSUBAR;                                                                GOS00960
```

```
FILE: READCG   PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

 READCG : PROCEDURE:                                                    READ00010
                                                                        READ00020
     DCL  1    IDN_AD                  (99)      EXT,                   READ00030
               2    IDN_ADR            CHAR      (4);                   READ00040
     DCL  OP1_PTR                      FIXED     (4)  EXT,              READ00050
          HEXCON                       CHAR      (2)  EXT,              READ00060
          MOCON                        CHAR      (9)  EXT,              READ00070
          CMTS                         CHAR      (45) EXT;              READ00080
     DCL  RITEOUT   ENTRY;                                              READ00090
                                                                        READ00100
     /* THIS ROUTINE READS A CONSTANT FROM THE DATA TABLE, ASSIGNS      READ00110
          IT TO A VARIABLE, AND INCREMENTS THE DATA TABLE POINTER */    READ00120
                                                                        READ00130
     /* LOAD ACCUMULATOR WITH DATA VALUE */                             READ00140
                                                                        READ00150
          HEXCON = 'A6';                                                READ00160
          MOCON = 'LDAA, X';                                            READ00170
          CMTS = 'LOAD DATA INTO ACCUMULATOR A USING';                  READ00180
          CALL RITEOUT;                                                 READ00190
          HEXCON = '00';                                                READ00200
          MOCON = ' ';                                                  READ00210
          CMTS = 'INDEXED ADDRESSING WITH ZERO OFFSET';                 READ00220
          CALL RITEOUT;                                                 READ00230
                                                                        READ00240
     /* STORE DATA IN APPROPRIATE VARIABLE NAME */                      READ00250
                                                                        READ00260
          HEXCON = '97';                                                READ00270
          MOCON = 'STAA';                                               READ00280
          CMTS = 'STORE DATA DIRECT INTO VARIABLE NAME';                READ00290
          CALL RITEOUT;                                                 READ00300
          HEXCON = SUBSTR (IDN_ADR(OP1_PTR), 3, 2);                     READ00310
          MOCON = ' ';                                                  READ00320
          CMTS = 'AT THIS ZERO PAGE ADDRESS';                           READ00330
          CALL RITEOUT;                                                 READ00340
                                                                        READ00350
     /* INCREMENT DATA TABLE POINTER */                                 READ00360
                                                                        READ00370
          HEXCON = '08';                                                READ00380
          MOCON = 'INX';                                                READ00390
          CMTS = 'INCREMENT INDEX REGISTER';                            READ00400
          CALL RITEOUT;                                                 READ00410
                                                                        READ00420
 END READCG;                                                            READ00430
```

```
FILE: STOPCG   PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

STOPCG : PROCEDURE:                                                    ST000010
                                                                       ST000020
    DCL  HEXCON                    CHAR      (2)  EXT.                ST000030
         MDCON                     CHAR      (9)  EXT.                ST000040
         CMTS                      CHAR      (45) EXT:                ST000050
    DCL  RITEOUT   ENTRY:                                              ST000060
                                                                       ST000070
    /* HALT PROGRAM EXECUTION */                                       ST000080
                                                                       ST000090
         HEXCON = '3E':                                                ST000100
         MDCON = 'WAI':                                                ST000110
         CMTS = 'HALT EXECUTION (WAIT FOR INTERRUPT)':                 ST000120
         CALL RITEOUT:                                                 ST000130
                                                                       ST000140
END STOPCG:                                                            ST000150
```

```
FILE: ENDCG    PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

ENDCG : PROCEDURE:                                                          END00010
                                                                            END00020
    DCL  HEXCON                    CHAR      (2)  EXT,                    END00030
         MDCON                     CHAR      (9)  EXT,                    END00040
         CMTS                      CHAR      (45) EXT;                    END00050
    DCL  RITEOUT   ENTRY;                                                   END00060
                                                                            END00070
    /* HALT PROGRAM EXECUTION */                                            END00080
                                                                            END00090
         HEXCON = '3E';                                                     END00100
         MDCON = 'WAI';                                                     END00110
         CMTS = 'HALT EXECUTION (WAIT FOR INTERRUPT)';                      END00120
         CALL RITEOUT;                                                      END00130
                                                                            END00140
END ENDCG;                                                                  END00150
```

```
FILE: MTPLYCG  PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

MTPLYCG : PROCEDURE;                                                    MTP00010
                                                                        MTP00020
    DCL  1    TMP_AD                (70)      EXT,                    MTP00030
              2    TMP_ADR          CHAR      (4);                    MTP00040
    DCL  1    LIT_AD                (500)     EXT,                    MTP00050
              2    LIT_ADR          CHAR      (4);                    MTP00060
    DCL  1    IDN_AD                (99)      EXT,                    MTP00070
              2    IDN_ADR          CHAR      (4);                    MTP00080
    DCL  OP1                        CHAR      (3)  EXT,               MTP00090
         OP1_PTR                    FIXED     (4)  EXT,               MTP00100
         OP2                        CHAR      (3)  EXT,               MTP00110
         OP2_PTR                    FIXED     (4)  EXT,               MTP00120
         HEXCON                     CHAR      (2)  EXT,               MTP00130
         MOCON                      CHAR      (9)  EXT,               MTP00140
         CMTS                       CHAR      (45) EXT,               MTP00150
         TMP_IDX                    FIXED     (2)  EXT;               MTP00160
    DCL  RITEOUT   ENTRY;                                               MTP00170
                                                                        MTP00180
    /* THIS ROUTINE MULTIPLIES TWO NUMBERS */                           MTP00190
                                                                        MTP00200
        HEXCON = '06';                                                  MTP00210
        MOCON = 'LDAB';                                                 MTP00220
        CMTS = 'LOAD THE B ACCUMULATOR WITH THE';                       MTP00230
        CALL RITEOUT;                                                   MTP00240
                                                                        MTP00250
        IF OP1 = 'LIT' THEN HEXCON = SUBSTR(LIT_ADR(OP1_PTR),3,2);      MTP00260
        IF OP1 = 'IDN' THEN HEXCON = SUBSTR(IDN_ADR(OP1_PTR),3,2);      MTP00270
        IF OP1 = 'TMP' THEN HEXCON = SUBSTR(TMP_ADR(OP1_PTR),3,2);      MTP00280
        MOCON = ' ';                                                    MTP00290
        CMTS = 'MULTIPLIER FROM THIS LOCATION';                         MTP00300
        CALL RITEOUT;                                                   MTP00310
                                                                        MTP00320
        HEXCON = '4F';                                                  MTP00330
        MOCON = 'CLRA';                                                 MTP00340
        CMTS = 'CLEAR ACCUMULATOR A';                                   MTP00350
        CALL RITEOUT;                                                   MTP00360
                                                                        MTP00370
        HEXCON = '97';                                                  MTP00380
        MOCON = 'STAA';                                                 MTP00390
        CMTS = 'STORE THE PRODUCT IN';                                  MTP00400
        CALL RITEOUT;                                                   MTP00410
                                                                        MTP00420
        HEXCON = SUBSTR(TMP_ADR(TMP_IDX),3,2);                          MTP00430
        MOCON = ' ';                                                    MTP00440
        CMTS = 'THIS TEMPORARY LOCATION';                               MTP00450
        CALL RITEOUT;                                                   MTP00460
                                                                        MTP00470
        HEXCON = '17';                                                  MTP00480
        MOCON = 'TBA';                                                  MTP00490
        CMTS = 'TRANSFER ACCUMULATOR B TO A';                           MTP00500
        CALL RITEOUT;                                                   MTP00510
                                                                        MTP00520
        HEXCON = '27';                                                  MTP00530
        MOCON = 'BEQ';                                                  MTP00540
        CMTS = 'IF THE MULTIPLIER IS ZERO, BRANCH';                     MTP00550
```

```
FILE: MTPLYCG  PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

        CALL RITEOUT;                                                    MTP00540
                                                                         MTP00570
        HEXCON = '08';                                                   MTP00580
        MOCON = ' ';                                                     MTP00590
        CMTS = 'TO THE NEXT ALGORITHM';                                  MTP00600
        CALL RITEOUT;                                                    MTP00610
                                                                         MTP00620
        HEXCON = '4A';                                                   MTP00630
        MOCON = 'DECA';                                                  MTP00640
        CMTS = 'OTHERWISE, DECREMENT THE MULTIPLIER';                    MTP00650
        CALL RITEOUT;                                                    MTP00660
                                                                         MTP00670
        HEXCON = '16';                                                   MTP00680
        MOCON = 'TAB';                                                   MTP00690
        CMTS = 'TRANSFER ACCUMULATOR A TO B';                            MTP00700
        CALL RITEOUT;                                                    MTP00710
                                                                         MTP00720
        HEXCON = '96';                                                   MTP00730
        MOCON = 'LDAA';                                                  MTP00740
        CMTS = 'LOAD THE ACCUMULATOR WITH THE';                          MTP00750
        CALL RITEOUT;                                                    MTP00760
                                                                         MTP00770
        HEXCON = SUBSTR(TMP_ADR(TMP_IDX),3,2);                           MTP00780
        MOCON = ' ';                                                     MTP00790
        CMTS = 'PRODUCT STORED IN THIS TEMPORARY LOCATION';              MTP00800
        CALL RITEOUT;                                                    MTP00810
                                                                         MTP00820
        HEXCON = '9B';                                                   MTP00830
        MOCON = 'ADDA';                                                  MTP00840
        CMTS = 'ADD TO THE PRODUCT';                                     MTP00850
        CALL RITEOUT;                                                    MTP00860
                                                                         MTP00870
        IF OP2 = 'LIT' THEN HEXCON = SUBSTR(LIT_ADR(OP2_PTR),3,2);       MTP00880
        IF OP2 = 'IDN' THEN HEXCON = SUBSTR(IDN_ADR(OP2_PTR),3,2);       MTP00890
        IF OP2 = 'TMP' THEN HEXCON = SUBSTR(TMP_ADR(OP2_PTR),3,2);       MTP00900
        MOCON = ' ';                                                     MTP00910
        CMTS = 'MULTIPLICAND AT THIS ADDRESS';                           MTP00920
        CALL RITEOUT;                                                    MTP00930
                                                                         MTP00940
        HEXCON = '20';                                                   MTP00950
        MOCON = 'BRA';                                                   MTP00960
        CMTS = 'BRANCH BACK TO BEGINNING OF';                            MTP00970
        CALL RITEOUT;                                                    MTP00980
                                                                         MTP00990
        HEXCON = 'F3';                                                   MTP01000
        MOCON = ' ';                                                     MTP01010
        CMTS = 'MULTIPLY ALGORITHM';                                     MTP01020
        CALL RITEOUT;                                                    MTP01030
                                                                         MTP01040
    /* INCREMENT TEMPORARY STORAGE INDEX */                              MTP01050
                                                                         MTP01060
        TMP_IDX = TMP_IDX + 1;                                           MTP01070
        RETURN;                                                          MTP01080
END MTPLYCG;                                                             MTP01090
                                                                         MTP01100
```

FILE: RESTRCG PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
RESTRCG : PROCEDURE;                                                    RES00010
                                                                        RES00020
    DCL  HEXCON                     CHAR      (2)  EXT,                 RES00030
         MOCON                      CHAR      (9)  EXT,                 RES00040
         CMTS                       CHAR      (45) EXT;                 RES00050
    DCL  RITEOUT   ENTRY;                                               RES00060
                                                                        RES00070
    /* THIS ROUTINE RESETS THE INDEX REGISTER TO THE TOP OF THE         RES00080
       DATA TABLE AT ADDRESS 0000 */                                    RES00090
                                                                        RES00100
       HEXCON = 'CE';                                                   RES00110
       MOCON = 'LDX';                                                   RES00120
       CMTS = 'RESET INDEX REGISTER TO';                                RES00130
       CALL RITEOUT;                                                    RES00140
       HEXCON = '00';                                                   RES00150
       MOCON = ' ';                                                     RES00160
       CMTS = 'TOP OF DATA TABLE STARTING';                             RES00170
       CALL RITEOUT;                                                    RES00180
       HEXCON = '00';                                                   RES00190
       MOCON = ' ';                                                     RES00200
       CMTS = 'AT ADDRESS 0000';                                        RES00210
       CALL RITEOUT;                                                    RES00220
       RETURN;                                                          RES00230
END RESTRCG;                                                            RES00240
```

FILE: EQUALCG PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
EQUALCG : PROCEDURE:                                                    EQU00010
                                                                        EQU00020
    DCL  1   TMP_AD                    (70)    EXT,                     EQU00030
             2   TMP_ADR               CHAR    (4):                     EQU00040
    DCL  1   LIT_AD                    (500)   EXT,                     EQU00050
             2   LIT_ADR               CHAR    (4):                     EQU00060
    DCL  1   IDN_AD                    (99)    EXT,                     EQU00070
             2   IDN_ADR               CHAR    (4):                     EQU00080
    DCL  OP1                           CHAR    (3)  EXT,                EQU00090
         OP1_PTR                       FIXED   (4)  EXT,                EQU00100
         OP2                           CHAR    (3)  EXT,                EQU00110
         OP2_PTR                       FIXED   (4)  EXT,                EQU00120
         HEXADR                        CHAR    (4)  EXT,                EQU00130
         HEXCON                        CHAR    (2)  EXT,                EQU00140
         MDCON                         CHAR    (9)  EXT,                EQU00150
         CMTS                          CHAR    (45) EXT:                EQU00160
    DCL  RITEOUT   ENTRY;                                               EQU00170
                                                                        EQU00180
    /* THIS ROUTINE ASSIGNS VALUES TO VARIABLES */                      EQU00190
                                                                        EQU00200
    /* WRITE LINE TO LOAD ACCUMULATOR */                                EQU00210
                                                                        EQU00220
        HEXCON = '96':                                                  EQU00230
        MDCON = 'LDAA':                                                 EQU00240
        CMTS = 'LOAD ACCUMULATOR A DIRECT':                             EQU00250
        CALL RITEOUT:                                                   EQU00260
                                                                        EQU00270
    /* FIND OPERAND 2 */                                                EQU00280
                                                                        EQU00290
        IF OP2 = 'TMP' THEN                                             EQU00300
            DO:                                                         EQU00310
                HEXCON = SUBSTR (TMP_ADR(OP2_PTR), 3, 2):               EQU00320
                MDCON = ' ':                                            EQU00330
                CMTS = 'WITH TEMP VALUE AT THIS ADDRESS':               EQU00340
            END:                                                        EQU00350
        IF OP2 = 'IDN' THEN                                             EQU00360
            DO:                                                         EQU00370
                HEXCON = SUBSTR (IDN_ADR(OP2_PTR), 3, 2):               EQU00380
                MDCON = ' ':                                            EQU00390
                CMTS = 'WITH IDENTIFIER VALUE AT THIS ADDRESS':         EQU00400
            END:                                                        EQU00410
        IF OP2 = 'LIT' THEN                                             EQU00420
            DO:                                                         EQU00430
                HEXCON = SUBSTR (LIT_ADR(OP2_PTR), 3, 2):               EQU00440
                MDCON = ' ':                                            EQU00450
                CMTS = 'WITH CONSTANT AT THIS ADDRESS':                 EQU00460
            END:                                                        EQU00470
        CALL RITEOUT:                                                   EQU00480
                                                                        EQU00490
    /* WRITE LINE TO STORE ACCUMULATOR IN MEMORY */                     EQU00500
                                                                        EQU00510
        HEXCON = '97':                                                  EQU00520
        MDCON = 'STAA':                                                 EQU00530
        CMTS = 'STORE ACCUMULATOR A DIRECT':                            EQU00540
        CALL RITEOUT:                                                   EQU00550
```

```
                                                                        EQU00560
   /* FIND OPERAND 1 */                                                 EQU00570
                                                                        EQU00580
      IF OP1 = 'IDN' THEN                                               EQU00590
         DO:                                                            EQU00600
            HEXCON = SUBSTR (IDN_ADR(OP1_PTR), 3, 2):                   EQU00610
            MDCON = ' ':                                                EQU00620
            CMTS = 'IN VARIABLE NAME AT THIS ADDRESS':                  EQU00630
         END:                                                           EQU00640
      IF OP1 = 'TMP' THEN                                               EQU00650
         DO:                                                            EQU00660
            HEXCON = SUBSTR (TMP_ADR(OP1_PTR), 3, 2):                   EQU00670
            MDCON = ' ':                                                EQU00680
            CMTS = 'IN TEMPORARY STORAGE AT THIS ADDRESS':              EQU00690
         END:                                                           EQU00700
      CALL RITEOUT:                                                     EQU00710
                                                                        EQU00720
   END EQUALCG:                                                         EQU00730
```

```
FILE: PLIISCG   PLIOPT   A          YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

PLIISCG : PROCEDURE;                                                    PLU00010
                                                                        PLU00020
   DCL  1   TMP_AD                (70)      EXT,                       PLU00030
            2   TMP_ADR           CHAR      (4);                       PLU00040
   DCL  1   LIT_AD                (500)     EXT,                       PLU00050
            2   LIT_ADR           CHAR      (4);                       PLU00060
   DCL  1   IDN_AD                (99)      EXT,                       PLU00070
            2   IDN_ADR           CHAR      (4);                       PLU00080
   DCL  OP1                       CHAR      (3)  EXT,                  PLU00090
        OP1_PTR                   FIXED     (4)  EXT,                  PLU00100
        OP2                       CHAR      (3)  EXT,                  PLU00110
        OP2_PTR                   FIXED     (4)  EXT,                  PLU00120
        HEXCON                    CHAR      (2)  EXT,                  PLU00130
        MNCON                     CHAR      (9)  EXT,                  PLU00140
        CMTS                      CHAR      (45) EXT,                  PLU00150
        TMP_IDX                   FIXED     (2)  EXT;                  PLU00160
   DCL  RITEOUT   ENTRY;                                                PLU00170
                                                                        PLU00180
   /* THIS ROUTINE ADDS TWO NUMBERS AND STORES THE RESULTS */           PLU00190
                                                                        PLU00200
   /* WRITE LINE TO LOAD OPERAND 1 INTO ACCUMULATOR */                  PLU00210
                                                                        PLU00220
      HEXCON = '96';                                                    PLU00230
      MNCON = 'LOAA';                                                   PLU00240
      CMTS = 'LOAD ACCUMULATOR A DIRECT';                               PLU00250
      CALL RITEOUT;                                                     PLU00260
      IF OP1 = 'LIT' THEN HEXCON = SUBSTR(LIT_ADR(OP1_PTR),3,2);        PLU00270
      IF OP1 = 'IDN' THEN HEXCON = SUBSTR(IDN_ADR(OP1_PTR),3,2);        PLU00280
      IF OP1 = 'TMP' THEN HEXCON = SUBSTR(TMP_ADR(OP1_PTR),3,2);        PLU00290
      MNCON = ' ';                                                      PLU00300
      CMTS = 'WITH FIRST OPERAND';                                      PLU00310
      CALL RITEOUT;                                                     PLU00320
                                                                        PLU00330
   /* ADD OPERAND TWO TO ACCUMULATOR */                                 PLU00340
                                                                        PLU00350
      HEXCON = '98';                                                    PLU00360
      MNCON = 'ADDA';                                                   PLU00370
      CMTS = 'ADD DIRECT TO ACCUMULATOR A';                             PLU00380
      CALL RITEOUT;                                                     PLU00390
      IF OP2 = 'LIT' THEN HEXCON = SUBSTR(LIT_ADR(OP2_PTR),3,2);        PLU00400
      IF OP2 = 'IDN' THEN HEXCON = SUBSTR(IDN_ADR(OP2_PTR),3,2);        PLU00410
      IF OP2 = 'TMP' THEN HEXCON = SUBSTR(TMP_ADR(OP2_PTR),3,2);        PLU00420
      MNCON = ' ';                                                      PLU00430
      CMTS = 'THE SECOND OPERAND';                                      PLU00440
      CALL RITEOUT;                                                     PLU00450
                                                                        PLU00460
   /* STORE THE SUM IN A TEMPORARY LOCATION */                          PLU00470
                                                                        PLU00480
      HEXCON = '97';                                                    PLU00490
      MNCON = 'STAA';                                                   PLU00500
      CMTS = 'STORE ACCUMULATOR A DIRECT';                              PLU00510
      CALL RITEOUT;                                                     PLU00520
      HEXCON = SUBSTR(TMP_ADR(TMP_IDX),3,2);                            PLU00530
      MNCON = ' ';                                                      PLU00540
      CMTS = 'IN THIS TEMPORARY LOCATION';                              PLU00550
```

```
        CALL RITEOUT;                                          PLU00560
                                                               PLU00570
    /* INCREMENT TEMPORARY STORAGE INDEX */                    PLU00580
                                                               PLU00590
        TMP_IDX = TMP_IDX + 1;                                 PLU00600
        RETURN;                                                PLU00610
                                                               PLU00620
END PLUSCG;                                                    PLU00630
```

```
FILE: MINUSCG  PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

 MINUSCG : PROCEDURE;                                                    MIN00010
                                                                         MIN00020
     DCL  1    TMP_AD                  (70)     EXT,                     MIN00030
               2    TMP_ADR            CHAR     (4);                     MIN00040
     DCL  1    LIT_AD                  (500)    EXT,                     MIN00050
               2    LIT_ADR            CHAR     (4);                     MIN00060
     DCL  1    IDN_AD                  (99)     EXT,                     MIN00070
               2    IDN_ADR            CHAR     (4);                     MIN00080
     DCL  OP1                          CHAR     (3)  EXT,                MIN00090
          OP1_PTR                      FIXED    (4)  EXT,                MIN00100
          OP2                          CHAR     (3)  EXT,                MIN00110
          OP2_PTR                      FIXED    (4)  EXT,                MIN00120
          HEXCON                       CHAR     (2)  EXT,                MIN00130
          MOCON                        CHAR     (9)  EXT,                MIN00140
          CMTS                         CHAR     (45) EXT,                MIN00150
          TMP_IDX                      FIXED    (2)  EXT;                MIN00160
     DCL  RITEOUT   ENTRY;                                               MIN00170
                                                                         MIN00180
     /* THIS ROUTINE SUBTRACTS TWO NUMBERS AND STORES THE RESULTS */      MIN00190
                                                                         MIN00200
     /* WRITE LINE TO LOAD OPERAND 1 INTO ACCUMULATOR */                 MIN00210
                                                                         MIN00220
          HEXCON = '96';                                                 MIN00230
          MOCON = 'LOAA';                                                MIN00240
          CMTS = 'LOAD ACCUMULATOR A DIRECT';                            MIN00250
          CALL RITEOUT;                                                  MIN00260
          IF OP1 = 'LIT' THEN HEXCON = SUBSTR(LIT_ADR(OP1_PTR),3,2);     MIN00270
          IF OP1 = 'IDN' THEN HEXCON = SUBSTR(IDN_ADR(OP1_PTR),3,2);     MIN00280
          IF OP1 = 'TMP' THEN HEXCON = SUBSTR(TMP_ADR(OP1_PTR),3,2);     MIN00290
          MOCON = ' ';                                                   MIN00300
          CMTS = 'WITH FIRST OPERAND';                                   MIN00310
          CALL RITEOUT;                                                  MIN00320
                                                                         MIN00330
     /* SUBTRACT OPERAND TWO FROM ACCUMULATOR */                         MIN00340
                                                                         MIN00350
          HEXCON = '90';                                                 MIN00360
          MOCON = 'SUBA';                                                MIN00370
          CMTS = 'SUBTRACT DIRECT FROM ACCUMULATOR A';                   MIN00380
          CALL RITEOUT;                                                  MIN00390
          IF OP2 = 'LIT' THEN HEXCON = SUBSTR(LIT_ADR(OP2_PTR),3,2);     MIN00400
          IF OP2 = 'IDN' THEN HEXCON = SUBSTR(IDN_ADR(OP2_PTR),3,2);     MIN00410
          IF OP2 = 'TMP' THEN HEXCON = SUBSTR(TMP_ADR(OP2_PTR),3,2);     MIN00420
          MOCON = ' ';                                                   MIN00430
          CMTS = 'THE SECOND OPERAND';                                   MIN00440
          CALL RITEOUT;                                                  MIN00450
                                                                         MIN00460
     /* STORE THE SUM IN A TEMPORARY LOCATION */                         MIN00470
                                                                         MIN00480
          HEXCON = '97';                                                 MIN00490
          MOCON = 'STAA';                                                MIN00500
          CMTS = 'STORE ACCUMULATOR A DIRECT';                           MIN00510
          CALL RITEOUT;                                                  MIN00520
          HEXCON = SUBSTR(TMP_ADR(TMP_IDX),3,2);                         MIN00530
          MOCON = ' ';                                                   MIN00540
          CMTS = 'IN THIS TEMPORARY LOCATION';                           MIN00550
```

```
        CALL RITEOUT;                                          MIN00560
                                                               MIN00570
    /* INCREMENT TEMPORARY STORAGE INDEX */                    MIN00580
                                                               MIN00590
        TMP_IDX = TMP_IDX + 1;                                 MIN00600
        RETURN;                                                MIN00610
                                                               MIN00620
END MINUSCG;                                                   MIN00630
```

FILE: DIVIDCG PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
DIVIDCG : PROCEDURE;                                                    DIV00010
                                                                        DIV00020
    DCL  1   TMP_AD                (70)    EXT,                         DIV00030
             2   TMP_ADR           CHAR    (4);                         DIV00040
    DCL  1   LIT_AD                (500)   EXT,                         DIV00050
             2   LIT_ADR           CHAR    (4);                         DIV00060
    DCL  1   IDN_AD                (99)    EXT,                         DIV00070
             2   IDN_ADR           CHAR    (4);                         DIV00080
    DCL  OP1                       CHAR    (3)   EXT,                   DIV00090
         OP1_PTR                   FIXED   (4)   EXT,                   DIV00100
         OP2                       CHAR    (3)   EXT,                   DIV00110
         OP2_PTR                   FIXED   (4)   EXT,                   DIV00120
         HEXCON                    CHAR    (2)   EXT,                   DIV00130
         MDCON                     CHAR    (9)   EXT,                   DIV00140
         CMTS                      CHAR    (45)  EXT,                   DIV00150
         TMP_IDX                   FIXED   (2)   EXT;                   DIV00160
    DCL  RITEOUT   ENTRY;                                               DIV00170
                                                                        DIV00180
    /* THIS ROUTINE PERFORMS DIVISION OF TWO NUMBERS */                 DIV00190
                                                                        DIV00200
         HEXCON = '06';                                                 DIV00210
         MDCON = 'LDAB';                                                DIV00220
         CMTS = 'LOAD THE B ACCUMULATOR WITH THE';                      DIV00230
         CALL RITEOUT;                                                  DIV00240
                                                                        DIV00250
         IF OP1 = 'LIT' THEN HEXCON = SUBSTR(LIT_ADR(OP1_PTR),3,2);     DIV00260
         IF OP1 = 'IDN' THEN HEXCON = SUBSTR(IDN_ADR(OP1_PTR),3,2);     DIV00270
         IF OP1 = 'TMP' THEN HEXCON = SUBSTR(TMP_ADR(OP1_PTR),3,2);     DIV00280
         MDCON = ' ';                                                   DIV00290
         CMTS = 'DIVIDEND FROM THIS LOCATION';                          DIV00300
         CALL RITEOUT;                                                  DIV00310
                                                                        DIV00320
         HEXCON = '4F';                                                 DIV00330
         MDCON = 'CLRA';                                                DIV00340
         CMTS = 'CLEAR ACCUMULATOR A';                                  DIV00350
         CALL RITEOUT;                                                  DIV00360
                                                                        DIV00370
         HEXCON = '97';                                                 DIV00380
         MDCON = 'STAA';                                                DIV00390
         CMTS = 'STORE THE QUOTIENT IN';                                DIV00400
         CALL RITEOUT;                                                  DIV00410
                                                                        DIV00420
         HEXCON = SUBSTR(TMP_ADR(TMP_IDX),3,2);                         DIV00430
         MDCON = ' ';                                                   DIV00440
         CMTS = 'THIS TEMPORARY LOCATION';                              DIV00450
         CALL RITEOUT;                                                  DIV00460
                                                                        DIV00470
         HEXCON = '17';                                                 DIV00480
         MDCON = 'TBA';                                                 DIV00490
         CMTS = 'TRANSFER ACCUMULATOR B TO A';                          DIV00500
         CALL RITEOUT;                                                  DIV00510
                                                                        DIV00520
         HEXCON = '90';                                                 DIV00530
         MDCON = 'SUBA';                                                DIV00540
         CMTS = 'SUBTRACT FROM THE DIVIDEND THE';                       DIV00550
```

```
FILE: DIVIDCG  PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

        CALL RITEOUT;                                                    DIV00560
                                                                         DIV00570
        IF OP2 = 'LIT' THEN HEXCON = SUBSTR(LIT_ADR(OP2_PTR),3,2);       DIV00580
        IF OP2 = 'IDN' THEN HEXCON = SUBSTR(IDN_ADR(OP2_PTR),3,2);       DIV00590
        IF OP2 = 'TMP' THEN HEXCON = SUBSTR(TMP_ADR(OP2_PTR),3,2);       DIV00600
        MDCON = ' ';                                                     DIV00610
        CMTS = 'DIVISOR AT THIS ADDRESS';                                DIV00620
        CALL RITEOUT;                                                    DIV00630
                                                                         DIV00640
        HEXCON = '2B';                                                   DIV00650
        MDCON = 'BMI';                                                   DIV00660
        CMTS = 'IF THE DIFFERENCE IS NEGATIVE,';                         DIV00670
        CALL RITEOUT;                                                    DIV00680
                                                                         DIV00690
        HEXCON = '06';                                                   DIV00700
        MDCON = ' ';                                                     DIV00710
        CMTS = 'BRANCH TO THE NEXT ALGORITHM';                           DIV00720
        CALL RITEOUT;                                                    DIV00730
                                                                         DIV00740
        HEXCON = '16';                                                   DIV00750
        MDCON = 'TAB';                                                   DIV00760
        CMTS = 'OTHERWISE, TRANSFER ACCUMULATOR A TO B';                 DIV00770
        CALL RITEOUT;                                                    DIV00780
                                                                         DIV00790
        HEXCON = '96';                                                   DIV00800
        MDCON = 'LDAa';                                                  DIV00810
        CMTS = 'LOAD THE ACCUMULATOR WITH THE';                          DIV00820
        CALL RITEOUT;                                                    DIV00830
                                                                         DIV00840
        HEXCON = SUBSTR(TMP_ADR(TMP_IDX),3,2);                           DIV00850
        MDCON = ' ';                                                     DIV00860
        CMTS = 'QUOTIENT STORED IN THIS TEMPORARY LOCATION';             DIV00870
        CALL RITEOUT;                                                    DIV00880
                                                                         DIV00890
        HEXCON = '4C';                                                   DIV00900
        MDCON = 'INCA';                                                  DIV00910
        CMTS = 'INCREMENT THE QUOTIENT';                                 DIV00920
        CALL RITEOUT;                                                    DIV00930
                                                                         DIV00940
        HEXCON = '20';                                                   DIV00950
        MDCON = 'BRA';                                                   DIV00960
        CMTS = 'BRANCH BACK TO BEGINNING OF';                            DIV00970
        CALL RITEOUT;                                                    DIV00980
                                                                         DIV00990
        HEXCON = 'F3';                                                   DIV01000
        MDCON = ' ';                                                     DIV01010
        CMTS = 'DIVIDE ALGORITHM';                                       DIV01020
        CALL RITEOUT;                                                    DIV01030
                                                                         DIV01040
    /* INCREMENT TEMPORARY STORAGE INDEX */                              DIV01050
                                                                         DIV01060
        TMP_IDX = TMP_IDX + 1;                                           DIV01070
        RETURN;                                                          DIV01080
                                                                         DIV01090
END DIVIDCG;                                                             DIV01100
```

```
FILE: EXPCG     PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

EXPCG : PROCEDURE:                                                          EXP00010
                                                                            EXP00020
    DCL  1    TMP_AD                (70)      EXT.                        EXP00030
              2    TMP_ADR          CHAR      (4);                        EXP00040
    DCL  1    LIT_AD                (500)     EXT.                        EXP00050
              2    LIT_ADR          CHAR      (4);                        EXP00060
    DCL 1     IDN_AD                (99)      EXT.                        EXP00070
              2    IDN_ADR          CHAR      (4);                        EXP00080
    DCL  OP1                        CHAR      (3)  EXT.                   EXP00090
         OP1_PTR                    FIXED     (4)  EXT.                   EXP00100
         OP2                        CHAR      (3)  EXT.                   EXP00110
         OP2_PTR                    FIXED     (4)  EXT.                   EXP00120
         HEXCON                     CHAR      (2)  EXT.                   EXP00130
         MOCON                      CHAR      (9)  EXT.                   EXP00140
         CMTS                       CHAR      (45) EXT.                   EXP00150
         HEXADR                     CHAR      (4)  EXT.                   EXP00160
         TMP_IDX                    FIXED     (2)  EXT.                   EXP00170
         SCRATCH1                   CHAR      (4);                        EXP00180
         SCRATCH2                   CHAR      (4);                        EXP00190
    DCL  RITEOUT   ENTRY;                                                   EXP00200
                                                                            EXP00210
    /* THIS ROUTINE RAISES A NUMBER TO A POWER */                           EXP00220
                                                                            EXP00230
         HEXCON = '20';                                                     EXP00240
         MOCON = 'BRA';                                                     EXP00250
         CMTS = 'CREATE A SCRATCHPAD';                                      EXP00260
         CALL RITEOUT;                                                      EXP00270
                                                                            EXP00280
         HEXCON = '01';                                                     EXP00290
         MOCON = ' ';                                                       EXP00300
         CMTS = 'LOCATION FOR';                                             EXP00310
         CALL RITEOUT;                                                      EXP00320
                                                                            EXP00330
         HEXCON = 'XX';                                                     EXP00340
         MOCON = ' ';                                                       EXP00350
         CMTS = 'EXPONENT';                                                 EXP00360
         CALL RITEOUT;                                                      EXP00370
         SCRATCH1 = HEXADR;                                                 EXP00380
                                                                            EXP00390
         HEXCON = '86';                                                     EXP00400
         MOCON = 'LDAA';                                                    EXP00410
         CMTS = 'SET RESULT OF';                                            EXP00420
         CALL RITEOUT;                                                      EXP00430
                                                                            EXP00440
         HEXCON = '01';                                                     EXP00450
         MOCON = ' ';                                                       EXP00460
         CMTS = 'EXPONENTIATION';                                           EXP00470
         CALL RITEOUT;                                                      EXP00480
                                                                            EXP00490
         HEXCON = '97';                                                     EXP00500
         MOCON = 'STAA';                                                    EXP00510
         CMTS = 'TO';                                                       EXP00520
         CALL RITEOUT;                                                      EXP00530
                                                                            EXP00540
         HEXCON = SUBSTR(TMP_ADR(TMP_IDX), 3, 2);                           EXP00550
```

```
        MDCON = ' ';                                                    EXP00560
        CMTS = 'ONE';                                                   EXP00570
        CALL RITEOUT;                                                   EXP00580
                                                                        EXP00590
        HEXCON = '96';                                                  EXP00600
        MDCON = 'LDAA';                                                 EXP00610
        CMTS = 'LOAD VALUE OF';                                         EXP00620
        CALL RITEOUT;                                                   EXP00630
                                                                        EXP00640
        IF OP2 = 'LIT' THEN HEXCON = SUBSTR(LIT_ADR(OP2_PTR), 3, 2);    EXP00650
        IF OP2 = 'ION' THEN HEXCON = SUBSTR(ION_ADR(OP2_PTR), 3, 2);    EXP00660
        IF OP2 = 'TMP' THEN HEXCON = SUBSTR(TMP_ADR(OP2_PTR), 3, 2);    EXP00670
        MDCON = ' ';                                                    EXP00680
        CMTS = 'EXPONENT';                                              EXP00690
        CALL RITEOUT;                                                   EXP00700
                                                                        EXP00710
        HEXCON = '87';                                                  EXP00720
        MDCON = 'STAA';                                                 EXP00730
        CMTS = 'INTO';                                                  EXP00740
        CALL RITEOUT;                                                   EXP00750
                                                                        EXP00760
        HEXCON = SUBSTR(SCRATCH1, 1, 2);                                EXP00770
        MDCON = ' ';                                                    EXP00780
        CMTS = 'SCRATCHPAD';                                            EXP00790
        CALL RITEOUT;                                                   EXP00800
                                                                        EXP00810
        HEXCON = SUBSTR(SCRATCH1, 3, 2);                                EXP00820
        MDCON = ' ';                                                    EXP00830
        CMTS = 'LOCATION';                                              EXP00840
        CALL RITEOUT;                                                   EXP00850
                                                                        EXP00860
        HEXCON = '27';                                                  EXP00870
        MDCON = 'BEQ';                                                  EXP00880
        CMTS = 'IF EXPONENT IS ZERO';                                   EXP00890
        CALL RITEOUT;                                                   EXP00900
                                                                        EXP00910
        HEXCON = '1F';                                                  EXP00920
        MDCON = ' ';                                                    EXP00930
        CMTS = 'GO ON TO NEXT ALGORITHM';                               EXP00940
        CALL RITEOUT;                                                   EXP00950
                                                                        EXP00960
        HEXCON = '20';                                                  EXP00970
        MDCON = 'BRA';                                                  EXP00980
        CMTS = 'CREATE A SCRATCHPAD';                                   EXP00990
        CALL RITEOUT;                                                   EXP01000
                                                                        EXP01010
        HEXCON = '01';                                                  EXP01020
        MDCON = ' ';                                                    EXP01030
        CMTS = 'LOCATION FOR';                                          EXP01040
        CALL RITEOUT;                                                   EXP01050
                                                                        EXP01060
        HEXCON = 'XX';                                                  EXP01070
        MDCON = ' ';                                                    EXP01080
        CMTS = 'PRODUCT';                                               EXP01090
        CALL RITEOUT;                                                   EXP01100
```

```
      SCRATCH2 = HEXADR;                                              EXP01110
                                                                      EXP01120
      HEXCON = '06';                                                  EXP01130
      MOCON = 'LDAB';                                                 EXP01140
      CMTS = 'LOAD ACCUMULATOR B WITH THE';                           EXP01150
      CALL RITEOUT;                                                   EXP01160
                                                                      EXP01170
      IF OP1 = 'LIT' THEN HEXCON = SUBSTR(LIT_ADR(OP1_PTR), 3, 2);    EXP01180
      IF OP1 = 'ION' THEN HEXCON = SUBSTR(ION_ADR(OP1_PTR), 3, 2);    EXP01190
      IF OP1 = 'TMP' THEN HEXCON = SUBSTR(TMP_ADR(OP1_PTR), 3, 2);    EXP01200
      MOCON = ' ';                                                    EXP01210
      CMTS = 'MULTIPLIER FROM THIS LOCATION';                         EXP01220
      CALL RITEOUT;                                                   EXP01230
                                                                      EXP01240
      HEXCON = '4F';                                                  EXP01250
      MOCON = 'CLRA';                                                 EXP01260
      CMTS = 'CLEAR ACCUMULATOR A';                                   EXP01270
      CALL RITEOUT;                                                   EXP01280
                                                                      EXP01290
      HEXCON = '87';                                                  EXP01300
      MOCON = 'STAA';                                                 EXP01310
      CMTS = 'STORE THE PRODUCT IN';                                  EXP01320
      CALL RITEOUT;                                                   EXP01330
                                                                      EXP01340
      HEXCON = SUBSTR(SCRATCH2, 1, 2);                                EXP01350
      MOCON = ' ';                                                    EXP01360
      CMTS = 'THIS';                                                  EXP01370
      CALL RITEOUT;                                                   EXP01380
                                                                      EXP01390
      HEXCON = SUBSTR(SCRATCH2, 3, 2);                                EXP01400
      MOCON = ' ';                                                    EXP01410
      CMTS = 'LOCATION';                                              EXP01420
      CALL RITEOUT;                                                   EXP01430
                                                                      EXP01440
      HEXCON = '17';                                                  EXP01450
      MOCON = 'TBA';                                                  EXP01460
      CMTS = 'TRANSFER ACCUMULATOR B TO A';                           EXP01470
      CALL RITEOUT;                                                   EXP01480
                                                                      EXP01490
      HEXCON = '27';                                                  EXP01500
      MOCON = 'BEQ';                                                  EXP01510
      CMTS = 'IF THE MULTIPLIER IS ZERO, EXIT';                       EXP01520
      CALL RITEOUT;                                                   EXP01530
                                                                      EXP01540
      HEXCON = '09';                                                  EXP01550
      MOCON = ' ';                                                    EXP01560
      CMTS = 'THE MULTIPLIER LOOP';                                   EXP01570
      CALL RITEOUT;                                                   EXP01580
                                                                      EXP01590
      HEXCON = '4A';                                                  EXP01600
      MOCON = 'DECA';                                                 EXP01610
      CMTS = 'OTHERWISE, DECREMENT THE MULTIPLIER';                   EXP01620
      CALL RITEOUT;                                                   EXP01630
                                                                      EXP01640
      HEXCON = '16';                                                  EXP01650
```

```
        MDCON = 'TAB':                                          EXP01640
        CMTS = 'TRANSFER ACCUMULATOR A TO B':                   EXP01670
        CALL RITEOUT:                                           EXP01680
                                                                EXP01690
        HEXCON = '86':                                          EXP01700
        MDCON = 'LDAA':                                         EXP01710
        CMTS = 'LOAD ACCUMULATOR':                              EXP01720
        CALL RITEOUT:                                           EXP01730
                                                                EXP01740
        HEXCON = SUBSTR(SCRATCH2, 1, 2):                        EXP01750
        MDCON = ' ':                                            EXP01760
        CMTS = 'WITH PRODUCT STORED':                           EXP01770
        CALL RITEOUT:                                           EXP01780
                                                                EXP01790
        HEXCON = SUBSTR(SCRATCH2, 3, 2):                        EXP01800
        MDCON = ' ':                                            EXP01810
        CMTS = 'IN THIS LOCATION':                              EXP01820
        CALL RITEOUT:                                           EXP01830
                                                                EXP01840
        HEXCON = '9B':                                          EXP01850
        MDCON = 'ADDA':                                         EXP01860
        CMTS = 'ADD TO THE PRODUCT THE':                        EXP01870
        CALL RITEOUT:                                           EXP01880
                                                                EXP01890
        HEXCON = SUBSTR(TMP_ADR(TMP_IDX), 3, 2):                EXP01900
        MDCON = ' ':                                            EXP01910
        CMTS = 'MULTIPLICAND AT THIS ADDRESS':                  EXP01920
        CALL RITEOUT:                                           EXP01930
                                                                EXP01940
        HEXCON = '20':                                          EXP01950
        MDCON = 'BRA':                                          EXP01960
        CMTS = 'BRANCH BACK TO BEGINNING':                      EXP01970
        CALL RITEOUT:                                           EXP01980
                                                                EXP01990
        HEXCON = 'F1':                                          EXP02000
        MDCON = ' ':                                            EXP02010
        CMTS = 'OF MULTIPLY ALGORITHM':                         EXP02020
        CALL RITEOUT:                                           EXP02030
                                                                EXP02040
        HEXCON = '86':                                          EXP02050
        MDCON ='LDAA':                                          EXP02060
        CMTS = 'LOAD ACCUMULATOR A':                            EXP02070
        CALL RITEOUT:                                           EXP02080
                                                                EXP02090
        HEXCON = SUBSTR(SCRATCH2, 1, 2):                        EXP02100
        MDCON = ' ':                                            EXP02110
        CMTS = 'WITH PRODUCT STORED':                           EXP02120
        CALL RITEOUT:                                           EXP02130
                                                                EXP02140
        HEXCON = SUBSTR(SCRATCH2, 3, 2):                        EXP02150
        MDCON = ' ':                                            EXP02160
        CMTS = 'IN THIS LOCATION':                              EXP02170
        CALL RITEOUT:                                           EXP02180
                                                                EXP02190
        HEXCON = '97':                                          EXP02200
```

```
FILE: EXPCG    PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

          MDCON = 'STAA':                                                  EXP02210
          CMTS = 'STORE RESULT IN THIS':                                   EXP02220
          CALL RITEOUT:                                                    EXP02230
                                                                           EXP02240
          HEXCON = SUBSTR(TMP_ADR(TMP_IDX), 3, 2):                         EXP02250
          MDCON = ' ':                                                     EXP02260
          CMTS = 'TEMPORARY STORAGE LOCATION':                             EXP02270
          CALL RITEOUT:                                                    EXP02280
                                                                           EXP02290
          HEXCON = '7A':                                                   EXP02300
          MDCON = 'DEC':                                                   EXP02310
          CMTS = 'DECREMENT':                                              EXP02320
          CALL RITEOUT:                                                    EXP02330
                                                                           EXP02340
          HEXCON = SUBSTR(SCRATCH1, 1, 2):                                 EXP02350
          MDCON = ' ':                                                     EXP02360
          CMTS = 'EXPONENT':                                               EXP02370
          CALL RITEOUT:                                                    EXP02380
                                                                           EXP02390
          HEXCON = SUBSTR(SCRATCH1, 3, 2):                                 EXP02400
          MDCON = ' ':                                                     EXP02410
          CMTS = 'LOCATION':                                               EXP02420
          CALL RITEOUT:                                                    EXP02430
                                                                           EXP02440
          HEXCON = '20':                                                   EXP02450
          MDCON = 'BRA':                                                   EXP02460
          CMTS = 'BRANCH BACK TO BEGINNING':                               EXP02470
          CALL RITEOUT:                                                    EXP02480
                                                                           EXP02490
          HEXCON = 'DF':                                                   EXP02500
          MDCON = ' ':                                                     EXP02510
          CMTS = 'OF EXPONENTIATION ALGORITHM':                            EXP02520
          CALL RITEOUT:                                                    EXP02530
                                                                           EXP02540
     /* INCREMENT TEMPORARY STORAGE INDEX */                               EXP02550
                                                                           EXP02560
          TMP_IDX = TMP_IDX + 1:                                           EXP02570
          RETURN:                                                          EXP02580
                                                                           EXP02590
  END EXPCG:                                                               EXP02600
```

```
FILE: SQRCG     PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

SQRCG : PROCEDURE:                                                          SQR00010
                                                                            SQR00020
   DCL  1    TMP_AD              (70)      EXT,                           SQR00030
             2    TMP_ADR        CHAR      (4):                           SQR00040
   DCL  1    LIT_AD              (500)     EXT,                           SQR00050
             2    LIT_ADR        CHAR      (4):                           SQR00060
   DCL 1     IDN_AD              (99)      EXT,                           SQR00070
             2    IDN_ADR        CHAR      (4):                           SQR00080
   DCL  OP1                      CHAR      (3)   EXT,                    SQR00090
        OP1_PTR                  FIXED     (4)   EXT,                    SQR00100
        OP2                      CHAR      (3)   EXT,                    SQR00110
        OP2_PTR                  FIXED     (4)   EXT,                    SQR00120
        HEXCON                   CHAR      (2)   EXT,                    SQR00130
        MOCON                    CHAR      (9)   EXT,                    SQR00140
        CMTS                     CHAR      (45)  EXT,                    SQR00150
        HEXADR                   CHAR      (4)   EXT,                    SQR00160
        TMP_IDX                  FIXED     (2)   EXT,                    SQR00170
        SCRATCH                  CHAR      (4):                          SQR00180
   DCL  RITEOUT   ENTRY:                                                  SQR00190
                                                                          SQR00200
   /* THIS ROUTINE COMPUTES SQUARE ROOT TO THE NEAREST WHOLE NUMBER */     SQR00210
                                                                          SQR00220
        HEXCON = '7F':                                                    SQR00230
        MOCON = 'CLR':                                                    SQR00240
        CMTS = 'INITIALIZE':                                              SQR00250
        CALL RITEOUT:                                                     SQR00260
                                                                          SQR00270
        HEXCON = SUBSTR(TMP_ADR(TMP_IDX), 1, 2):                          SQR00280
        MOCON = ' ':                                                      SQR00290
        CMTS = 'ANSWER':                                                  SQR00300
        CALL RITEOUT:                                                     SQR00310
                                                                          SQR00320
        HEXCON = SUBSTR(TMP_ADR(TMP_IDX), 3, 2):                          SQR00330
        MOCON = ' ':                                                      SQR00340
        CMTS = 'AT ZERO':                                                 SQR00350
        CALL RITEOUT:                                                     SQR00360
                                                                          SQR00370
        HEXCON = '20':                                                    SQR00380
        MOCON = 'BRA':                                                    SQR00390
        CMTS = 'CREATE A SCRATCHPAD':                                     SQR00400
        CALL RITEOUT:                                                     SQR00410
                                                                          SQR00420
        HEXCON = '01':                                                    SQR00430
        MOCON = ' ':                                                      SQR00440
        CMTS = 'LOCATION FOR':                                            SQR00450
        CALL RITEOUT:                                                     SQR00460
                                                                          SQR00470
        HEXCON = 'XX':                                                    SQR00480
        MOCON = ' ':                                                      SQR00490
        CMTS = 'SQUARE OF ANSWER':                                        SQR00500
        CALL RITEOUT:                                                     SQR00510
        SCRATCH = HEXADR:                                                 SQR00520
                                                                          SQR00530
        HEXCON = 'D6':                                                    SQR00540
        MOCON = 'LDAB':                                                   SQR00550
```

```
LE: SORCG    PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

      CMTS = 'LOAD THE B ACCUMULATOR WITH':                          SOR00560
      CALL RITEOUT:                                                  SOR00570
                                                                     SOR00580
      HEXCON = SUBSTR(TMP_ADR(TMP_IDX), 3, 2):                       SOR00590
      MOCON = ' ':                                                   SOR00600
      CMTS = 'THE MULTIPLIER FROM THIS LOCATION':                    SOR00610
      CALL RITEOUT:                                                  SOR00620
                                                                     SOR00630
      HEXCON = '4F':                                                 SOR00640
      MOCON = 'CLRA':                                                SOR00650
      CMTS = 'CLEAR ACCUMULATOR A':                                  SOR00660
      CALL RITEOUT:                                                  SOR00670
                                                                     SOR00680
      HEXCON = '87':                                                 SOR00690
      MOCON = 'STAA':                                                SOR00700
      CMTS = 'STORE THE PRODUCT IN':                                 SOR00710
      CALL RITEOUT:                                                  SOR00720
                                                                     SOR00730
      HEXCON = SUBSTR(SCRATCH, 1, 2):                                SOR00740
      MOCON = ' ':                                                   SOR00750
      CMTS = 'THIS SCRATCHPAD':                                      SOR00760
      CALL RITEOUT:                                                  SOR00770
                                                                     SOR00780
      HEXCON = SUBSTR(SCRATCH, 3, 2):                                SOR00790
      MOCON = ' ':                                                   SOR00800
      CMTS = 'LOCATION':                                             SOR00810
      CALL RITEOUT:                                                  SOR00820
                                                                     SOR00830
      HEXCON = '17':                                                 SOR00840
      MOCON = 'TBA':                                                 SOR00850
      CMTS = 'TRANSFER ACCUMULATOR B TO A':                          SOR00860
      CALL RITEOUT:                                                  SOR00870
                                                                     SOR00880
      HEXCON = '27':                                                 SOR00890
      MOCON = 'BEQ':                                                 SOR00900
      CMTS = 'IF THE MULTIPLIER IS ZERO,':                           SOR00910
      CALL RITEOUT:                                                  SOR00920
                                                                     SOR00930
      HEXCON = '09':                                                 SOR00940
      MOCON = ' ':                                                   SOR00950
      CMTS = 'CONTINUE WITH ALGORITHM':                              SOR00960
      CALL RITEOUT:                                                  SOR00970
                                                                     SOR00980
      HEXCON = '4A':                                                 SOR00990
      MOCON = 'DECA':                                                SOR01000
      CMTS = 'OTHERWISE, DECREMENT MULTIPLIER':                      SOR01010
      CALL RITEOUT:                                                  SOR01020
                                                                     SOR01030
      HEXCON = '16':                                                 SOR01040
      MOCON = 'TAB':                                                 SOR01050
      CMTS = 'TRANSFER ACCUMULATOR A TO B':                          SOR01060
      CALL RITEOUT:                                                  SOR01070
                                                                     SOR01080
      HEXCON = '86':                                                 SOR01090
      MOCON = 'LDAA':                                                SOR01100
```

```
FILE: SORCG     PLIOPT    A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

          CMTS = 'LOAD THE ACCUMULATOR WITH':                                  SOR01110
          CALL RITEOUT:                                                        SOR01120
                                                                               SOR01130
          HEXCON = SUBSTR(SCRATCH, 1, 2):                                      SOR01140
          MOCON = ' ':                                                         SOR01150
          CMTS = 'THE PRODUCT STORED IN':                                      SOR01160
          CALL RITEOUT:                                                        SOR01170
                                                                               SOR01180
          HEXCON = SUBSTR(SCRATCH, 3, 2):                                      SOR01190
          MOCON = ' ':                                                         SOR01200
          CMTS = 'THIS SCRATCHPAD LOCATION':                                   SOR01210
          CALL RITEOUT:                                                        SOR01220
                                                                               SOR01230
          HEXCON = '98':                                                       SOR01240
          MOCON = 'ADDA':                                                      SOR01250
          CMTS = 'ADD TO THE PRODUCT THE':                                     SOR01260
          CALL RITEOUT:                                                        SOR01270
                                                                               SOR01280
          HEXCON = SUBSTR(TMP_ADR(TMP_IDX), 3, 2):                             SOR01290
          MOCON = ' ':                                                         SOR01300
          CMTS = 'MULTIPLICAND AT THIS ADDRESS':                               SOR01310
          CALL RITEOUT:                                                        SOR01320
                                                                               SOR01330
          HEXCON = '20':                                                       SOR01340
          MOCON = 'BRA':                                                       SOR01350
          CMTS ='BRANCH BACK TO BEGINNING':                                    SOR01360
          CALL RITEOUT:                                                        SOR01370
                                                                               SOR01380
          HEXCON = 'F1':                                                       SOR01390
          MOCON = ' ':                                                         SOR01400
          CMTS = 'OF MULTIPLY SEQUENCE':                                       SOR01410
          CALL RITEOUT:                                                        SOR01420
                                                                               SOR01430
          HEXCON = '86':                                                       SOR01440
          MOCON = 'LDAA':                                                      SOR01450
          CMTS = 'LOAD ANSWER':                                                SOR01460
          CALL RITEOUT:                                                        SOR01470
                                                                               SOR01480
          HEXCON = SUBSTR(SCRATCH, 1, 2):                                      SOR01490
          MOCON = ' ':                                                         SOR01500
          CMTS = 'SQUARED INTO':                                               SOR01510
          CALL RITEOUT:                                                        SOR01520
                                                                               SOR01530
          HEXCON = SUBSTR(SCRATCH, 3, 2):                                      SOR01540
          MOCON = ' ':                                                         SOR01550
          CMTS = 'ACCUMULATOR A':                                              SOR01560
          CALL RITEOUT:                                                        SOR01570
                                                                               SOR01580
          HEXCON = '90':                                                       SOR01590
          MOCON = 'SUBA':                                                      SOR01600
          CMTS = 'SUBTRACT THE ORIGINAL':                                      SOR01610
          CALL RITEOUT:                                                        SOR01620
                                                                               SOR01630
          IF OP1 = 'LIT' THEN HEXCON = SUBSTR(LIT_ADR(OP1_PTR), 3, 2):         SOR01640
          IF OP1 = 'IDN' THEN HEXCON = SUBSTR(IDN_ADR(OP1_PTR), 3, 2):         SOR01650
```

```
FILE: SORCG    PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

        IF OP1 = 'TMP' THEN HEXCON = SUBSTR(TMP_ADR(OP1_PTR), 3, 2);  SOR01660
        MOCON = ' ';                                                  SOR01670
        CMTS = 'NUMBER FROM ACCUMULATOR A';                           SOR01680
        CALL RITEOUT;                                                 SOR01690
                                                                      SOR01700
        HEXCON = '2C';                                                SOR01710
        MOCON = 'BGE';                                                SOR01720
        CMTS = 'IF ANSWER SQUARED IS GREATER THAN';                   SOR01730
        CALL RITEOUT;                                                 SOR01740
                                                                      SOR01750
        HEXCON = '05';                                                SOR01760
        MOCON = ' ';                                                  SOR01770
        CMTS = 'OR EQUAL TO ARGUMENT, GO TO NEXT ALGORITHM';          SOR01780
        CALL RITEOUT;                                                 SOR01790
                                                                      SOR01800
        HEXCON = '7C';                                                SOR01810
        MOCON = 'INC';                                                SOR01820
        CMTS = 'OTHERWISE, INCREMENT THE';                            SOR01830
        CALL RITEOUT;                                                 SOR01840
                                                                      SOR01850
        HEXCON = SUBSTR(TMP_ADR(TMP_IDX), 1, 2);                      SOR01860
        MOCON = ' ';                                                  SOR01870
        CMTS = 'ANSWER AND';                                          SOR01880
        CALL RITEOUT;                                                 SOR01890
                                                                      SOR01900
        HEXCON = SUBSTR(TMP_ADR(TMP_IDX), 3, 2);                      SOR01910
        MOCON = ' ';                                                  SOR01920
        CMTS = 'RETURN TO';                                           SOR01930
        CALL RITEOUT;                                                 SOR01940
                                                                      SOR01950
        HEXCON = '20';                                                SOR01960
        MOCON = 'BRA';                                                SOR01970
        CMTS = 'SQUARING';                                            SOR01980
        CALL RITEOUT;                                                 SOR01990
                                                                      SOR02000
        HEXCON = 'E2';                                                SOR02010
        MOCON = ' ';                                                  SOR02020
        CMTS = 'ROUTINE';                                             SOR02030
        CALL RITEOUT;                                                 SOR02040
                                                                      SOR02050
    /* INCREMENT TEMPORARY STORAGE INDEX */                           SOR02060
                                                                      SOR02070
        TMP_IDX = TMP_IDX + 1;                                        SOR02080
                                                                      SOR02090
        RETURN;                                                       SOR02100
                                                                      SOR02110
END SORCG;                                                            SOR02120
```

```
FILE: RDINCG   PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

RDINCG : PROCEDURE:                                                    RDI00010
                                                                       RDI00020
    DCL  1    LIT_TBL              (500)     EXT.                      RDI00030
              2    LT_NTRY         CHAR      (9):                      RDI00040
    DCL  1    IDN_AD               (99)      EXT.                      RDI00050
              2    IDN_ADR         CHAR      (4):                      RDI00060
    DCL  HEXCON                    CHAR      (2)   EXT.               RDI00070
         MDCON                     CHAR      (9)   EXT.               RDI00080
         CMTS                      CHAR      (45)  EXT.               RDI00090
         OP1_PTR                   FIXED     (4)   EXT.               RDI00100
         OP2_PTR                   FIXED     (4)   EXT:              RDI00110
    DCL  RITEOUT   ENTRY:                                              RDI00120
                                                                       RDI00130
    /* THIS ROUTINE READS DATA FROM AN INPUT DEVICE */                 RDI00140
                                                                       RDI00150
         HEXCON = '86':                                                RDI00160
         MDCON = 'LDAA':                                               RDI00170
         CMTS = 'LOAD DATA FROM INPUT':                                RDI00180
         CALL RITEOUT:                                                 RDI00190
                                                                       RDI00200
         HEXCON = SUBSTR(LT_NTRY(OP2_PTR), 1, 2):                      RDI00210
         MDCON = ' ':                                                  RDI00220
         CMTS = 'DEVICE AT THIS ADDRESS':                              RDI00230
         CALL RITEOUT:                                                 RDI00240
                                                                       RDI00250
         HEXCON = SUBSTR(LT_NTRY(OP2_PTR), 3, 2):                      RDI00260
         MDCON = ' ':                                                  RDI00270
         CMTS = 'INTO ACCUMULATOR A, THEN':                            RDI00280
         CALL RITEOUT:                                                 RDI00290
                                                                       RDI00300
         HEXCON = '97':                                                RDI00310
         MDCON = 'STAA':                                               RDI00320
         CMTS = 'STORE DATA IN VARIABLE':                              RDI00330
         CALL RITEOUT:                                                 RDI00340
                                                                       RDI00350
         HEXCON = SUBSTR(IDN_ADR(OP1_PTR), 3, 2):                      RDI00360
         MDCON = ' ':                                                  RDI00370
         CMTS = 'NAME AT THIS ADDRESS':                                RDI00380
         CALL RITEOUT:                                                 RDI00390
                                                                       RDI00400
         RETURN:                                                       RDI00410
                                                                       RDI00420
END RDINCG:                                                            RDI00430
```

FILE: WTOUTCG PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
WTOUTCG : PROCEDURE;                                                    WT000010
                                                                        WT000020
    DCL  1   LIT_TBL                (500)    EXT,                      WT000030
             2   LT_NTRY            CHAR     (9);                      WT000040
    DCL  1   IDN_AD                 (99)     EXT,                      WT000050
             2   IDN_ADR            CHAR     (4);                      WT000060
    DCL  HEXCON                     CHAR     (2)  EXT,                 WT000070
         MDCON                      CHAR     (9)  EXT,                 WT000080
         CMTS                       CHAR     (45) EXT,                 WT000090
         OP1_PTR                    FIXED    (4)  EXT,                 WT000100
         OP2_PTR                    FIXED    (4)  EXT;                 WT000110
    DCL  RITEOUT   ENTRY;                                               WT000120
                                                                        WT000130
    /* THIS ROUTINE WRITES DATA TO AN OUTPUT DEVICE */                  WT000140
                                                                        WT000150
         HEXCON = '96';                                                 WT000160
         MDCON = 'LDAA';                                                WT000170
         CMTS = 'LOAD DATA STORED IN VARIABLE';                         WT000180
         CALL RITEOUT;                                                  WT000190
                                                                        WT000200
         HEXCON = SUBSTR(IDN_ADR(OP1_PTR), 3, 2);                       WT000210
         MDCON = ' ';                                                   WT000220
         CMTS = 'AT THIS ADDRESS INTO ACCUMULATOR A.';                  WT000230
         CALL RITEOUT;                                                  WT000240
                                                                        WT000250
         HEXCON = 'B7';                                                 WT000260
         MDCON = 'STAA';                                                WT000270
         CMTS = 'THEN WRITE IT TO';                                     WT000280
         CALL RITEOUT;                                                  WT000290
                                                                        WT000300
         HEXCON = SUBSTR(LT_NTRY(OP2_PTR), 1, 2);                       WT000310
         MDCON = ' ';                                                   WT000320
         CMTS = 'THE OUTPUT DEVICE RESIDING';                           WT000330
         CALL RITEOUT;                                                  WT000340
                                                                        WT000350
         HEXCON = SUBSTR(LT_NTRY(OP2_PTR), 3, 2);                       WT000360
         MDCON = ' ';                                                   WT000370
         CMTS = 'AT THIS HEX ADDRESS';                                  WT000380
         CALL RITEOUT;                                                  WT000390
                                                                        WT000400
         RETURN;                                                        WT000410
                                                                        WT000420
END WTOUTCG;                                                            WT000430
```

```
FILE: GOSUBCG  PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

GOSUBCG : PROCEDURE:                                                    GOS00010
                                                                        GOS00020
    DCL  1    GOTBL                    (100)     EXT,                   GOS00030
              2    LAB_PTR             FIXED     (4),                   GOS00040
              2    GO_ADR              FIXED     (5),                   GOS00050
              2    BRANCH              CHAR      (5),                   GOS00060
              2    DESTN               FIXED     (4),                   GOS00070
              2    OFFSET              FIXED     (4);                   GOS00080
    DCL  OP1_PTR                      FIXED     (4)  EXT,              GOS00090
         OP2_PTR                      FIXED     (4)  EXT,              GOS00100
         HEXCON                       CHAR      (2)  EXT,              GOS00110
         MDCON                        CHAR      (9)  EXT,              GOS00120
         CMTS                         CHAR      (45) EXT,              GOS00130
         DECADR                       FIXED     (5)  EXT,              GOS00140
         J                            FIXED     (1),                   GOS00150
         GT_IDX                       FIXED     (4)  EXT,              GOS00160
         LIN_PTR                      FIXED     (4)  EXT;              GOS00170
    DCL  RITEOUT   ENTRY;                                               GOS00180
                                                                        GOS00190
    /* THIS ROUTINE STORES GOSUB DESTINATIONS AND OFFSETS FOR           GOS00200
         PROCESSING AFTER COMPLETION OF COMPILATION */                  GOS00210
                                                                        GOS00220
    /* STORE INFORMATION IN GO TABLE */                                 GOS00230
                                                                        GOS00240
         LAB_PTR(GT_IDX) = LIN_PTR;                                     GOS00250
         GO_ADR(GT_IDX) = DECADR;                                       GOS00260
         BRANCH(GT_IDX) = 'GOSUB';                                      GOS00270
         DESTN(GT_IDX) = OP1_PTR;                                       GOS00280
         OFFSET(GT_IDX) = OP2_PTR;                                      GOS00290
                                                                        GOS00300
    /* RESERVE CORE FOR JUMP INSTRUCTION LATER */                       GOS00310
                                                                        GOS00320
         DO J=1 TO 3 BY 1;                                              GOS00330
              HEXCON = 'XX';                                            GOS00340
              MDCON = 'XXXXXXXXX';                                      GOS00350
              CMTS = 'BRANCH INSTRUCTION - TO BE FILLED IN LATER';      GOS00360
              CALL RITEOUT;                                             GOS00370
         END;                                                           GOS00380
                                                                        GOS00390
         GT_IDX = GT_IDX + 1;                                           GOS00400
                                                                        GOS00410
         RETURN;                                                        GOS00420
END GOSUBCG;                                                            GOS00430
```

FILE: GOTOCG PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
GOTOCG : PROCEDURE:                                                    GOT00010
                                                                       GOT00020
    DCL  1   GOTBL                    (100)   EXT.                     GOT00030
             2   LAB_PTR              FIXED   (4).                     GOT00040
             2   GO_ADR               FIXED   (5).                     GOT00050
             2   BRANCH               CHAR    (5).                     GOT00060
             2   DESTN                FIXED   (4).                     GOT00070
             2   OFFSET               FIXED   (4):                     GOT00080
    DCL  OP1_PTR                      FIXED   (4)  EXT.                GOT00090
         OP2_PTR                      FIXED   (4)  EXT.                GOT00100
         HEXCON                       CHAR    (2)  EXT.                GOT00110
         MOCON                        CHAR    (9)  EXT.                GOT00120
         CMTS                         CHAR    (45) EXT.                GOT00130
         DECADR                       FIXED   (5)  EXT.                GOT00140
         J                            FIXED   (1).                     GOT00150
         GT_IDX                       FIXED   (4)  EXT.                GOT00160
         LIN_PTR                      FIXED   (4)  EXT:                GOT00170
    DCL  RITEOUT   ENTRY:                                              GOT00180
                                                                       GOT00190
    /* THIS ROUTINE STORES GOTO DESTINATIONS AND OFFSETS FOR           GOT00200
         PROCESSING AFTER COMPLETION OF COMPILATION */                 GOT00210
                                                                       GOT00220
    /* STORE INFORMATION IN GO TABLE */                                GOT00230
                                                                       GOT00240
         LAB_PTR(GT_IDX) = LIN_PTR:                                    GOT00250
         GO_ADR(GT_IDX) = DECADR:                                      GOT00260
         BRANCH(GT_IDX) = 'GOTO ':                                     GOT00270
         DESTN(GT_IDX) = OP1_PTR:                                      GOT00280
         OFFSET(GT_IDX) = OP2_PTR:                                     GOT00290
                                                                       GOT00300
    /* RESERVE CORE FOR JUMP INSTRUCTION LATER */                      GOT00310
                                                                       GOT00320
         DO J=1 TO 3 BY 1:                                             GOT00330
              HEXCON = 'XX':                                           GOT00340
              MOCON = 'XXXXXXXXX':                                     GOT00350
              CMTS = 'BRANCH INSTRUCTION - TO BE FILLED IN LATER':     GOT00360
              CALL RITEOUT:                                            GOT00370
         END:                                                          GOT00380
                                                                       GOT00390
         GT_IDX = GT_IDX + 1:                                          GOT00400
                                                                       GOT00410
         RETURN:                                                       GOT00420
END GOTOCG:                                                            GOT00430
```

FILE: RTRNCG PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
RTRNCG : PROCEDURE:                                                    RTR00010
                                                                       RTR00020
    DCL HEXCON                  CHAR      (2)  EXT.                    RTR00030
        MOCON                   CHAR      (9)  EXT.                    RTR00040
        CMTS                    CHAR      (45) EXT:                    RTR00050
    DCL RITEOUT   ENTRY:                                               RTR00060
                                                                       RTR00070
        HEXCON = '39':                                                 RTR00080
        MOCON = 'RTS':                                                 RTR00090
        CMTS = 'RETURN FROM SUBROUTINE':                               RTR00100
        CALL RITEOUT:                                                  RTR00110
                                                                       RTR00120
        RETURN:                                                        RTR00130
END RTRNCG:                                                            RTR00140
```

```
FILE: BRANCHG  PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

BRANCHG : PROCEDURE:                                                          BRA00010
                                                                              BRA00020
    DCL  1   LBL_AD                    (500)    EXT,                          BRA00030
             2    LBL_PTR              FIXED    (4),                          BRA00040
             2    LBL_ADR              CHAR     (4);                          BRA00050
    DCL  1   GOTBL                     (100)    EXT,                          BRA00060
             2    LBR_PTR              FIXED    (4),                          BRA00070
             2    GO_ADR               FIXED    (5),                          BRA00080
             2    BRANCH               CHAR     (5),                          BRA00090
             2    DESTN                FIXED    (4),                          BRA00100
             2    OFFSET               FIXED    (4);                          BRA00110
    DCL  HEXCON                        CHAR     (2)   EXT,                    BRA00120
         MOCON                         CHAR     (9)   EXT,                    BRA00130
         CMTS                          CHAR     (45)  EXT,                    BRA00140
         DECADR                        FIXED    (5)   EXT,                    BRA00150
         GT_IDX                        FIXED    (4)   EXT,                    BRA00160
         DECDEST                       FIXED    (5)   EXT,                    BRA00170
         NEXT_ADR                      CHAR     (2),                          BRA00180
         DECADR_SAVE                   FIXED    (5),                          BRA00190
         HEXADR                        CHAR     (4)   EXT,                    BRA00200
         LBL_IDX                       FIXED    (4)   EXT;                    BRA00210
    DCL  RITEOUT   ENTRY;                                                     BRA00220
    DCL  DCHXCO    ENTRY;                                                     BRA00230
    DCL  HXDCCON   ENTRY;                                                     BRA00240
                                                                              BRA00250
    /* THIS ROUTINE ASSIGNS DESTINATIONS TO BRANCH STATEMENTS */              BRA00260
                                                                              BRA00270
        GT_IDX = 0;                                                           BRA00280
                                                                              BRA00290
    /* READ A GO TABLE ENTRY */                                               BRA00300
                                                                              BRA00310
B001:   GT_IDX = GT_IDX + 1;                                                  BRA00320
                                                                              BRA00330
    /* IF BRANCH ENTRY IS BLANK, RETURN TO MAIN */                            BRA00340
                                                                              BRA00350
        IF BRANCH(GT_IDX) = ' ' THEN RETURN;                                  BRA00360
        IF BRANCH(GT_IDX) = 'GOSUB' THEN                                      BRA00370
            DO;                                                               BRA00380
                DECADR = GO_ADR(GT_IDX);                                      BRA00390
                HEXCON = 'BD';                                                BRA00400
                MOCON = 'JSR';                                                BRA00410
                CMTS = 'JUMP TO SUBROUTINE';                                  BRA00420
                CALL RITEOUT;                                                 BRA00430
                                                                              BRA00440
                LBL_IDX = 1;                                                  BRA00450
                                                                              BRA00460
                DO WHILE (DESTN(GT_IDX)¬=LBL_PTR(LBL_IDX));                   BRA00470
                    LBL_IDX = LBL_IDX + 1;                                    BRA00480
                END;                                                          BRA00490
                                                                              BRA00500
                HEXCON = SUBSTR(LBL_ADR(LBL_IDX), 1, 2);                      BRA00510
                MOCON = ' ';                                                  BRA00520
                CMTS = 'BEGINNING AT THIS';                                   BRA00530
                CALL RITEOUT;                                                 BRA00540
                HEXCON = SUBSTR(LBL_ADR(LBL_IDX), 3, 2);                      BRA00550
```

```
FILE: BRANCHG  PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

                    MOCON = ' ';                                          BRA00560
                    CMTS = 'MEMORY LOCATION';                             BRA00570
                    CALL RITEOUT;                                         BRA00580
                    GO TO A001;                                           BRA00590
               END;                                                       BRA00600
                                                                          BRA00610
          IF OFFSET(GT_IDX) = 0 THEN                                      BRA00620
               DO;                                                        BRA00630
                    DECADR = GO_ADR(GT_IDX);                              BRA00640
                    HEXCON = '7E';                                        BRA00650
                    MOCON = 'JMP';                                        BRA00660
                    CMTS = 'JUMP TO';                                     BRA00670
                    CALL RITEOUT;                                         BRA00680
                                                                          BRA00690
                    LBL_IDX = 1;                                          BRA00700
                                                                          BRA00710
                    DO WHILE (DESTN(GT_IDX)¬=LBL_PTR(LBL_IDX));           BRA00720
                         LBL_IDX = LBL_IDX + 1;                           BRA00730
                    END;                                                  BRA00740
                                                                          BRA00750
                    HEXCON = SUBSTR(LBL_ADR(LBL_IDX), 1, 2);              BRA00760
                    MOCON = ' ';                                          BRA00770
                    CMTS = 'THIS';                                        BRA00780
                    CALL RITEOUT;                                         BRA00790
                    HEXCON = SUBSTR(LBL_ADR(LBL_IDX), 3, 2);              BRA00800
                    MOCON = ' ';                                          BRA00810
                    CMTS = 'HEX ADDRESS';                                 BRA00820
                    CALL RITEOUT;                                         BRA00830
                    GO TO A001;                                           BRA00840
               END;                                                       BRA00850
                                                                          BRA00860
          DECADR = GO_ADR(GT_IDX);                                        BRA00870
          HEXCON = '7E';                                                  BRA00880
          MOCON = 'JMP';                                                  BRA00890
          CMTS = 'JUMP TO';                                               BRA00900
          CALL RITEOUT;                                                   BRA00910
                                                                          BRA00920
          LBL_IDX = 1;                                                    BRA00930
                                                                          BRA00940
          DO WHILE (DESTN(GT_IDX)¬=LBL_PTR(LBL_IDX));                     BRA00950
               LBL_IDX = LBL_IDX + 1;                                     BRA00960
          END;                                                            BRA00970
                                                                          BRA00980
          DECADR_SAVE = DECADR;                                           BRA00990
          CALL HXDCCON;                                                   BRA01000
          DECADR = DECDEST + 4;                                           BRA01010
          CALL DCHXCO;                                                    BRA01020
          HEXCON = SUBSTR(HEXADR, 1, 2);                                  BRA01030
          NEXT_ADR = SUBSTR(HEXADR, 3, 2);                                BRA01040
          DECADR = DECADR_SAVE;                                           BRA01050
          MOCON = ' ';                                                    BRA01060
          CMTS = 'THIS';                                                  BRA01070
          CALL RITEOUT;                                                   BRA01080
          HEXCON = NEXT_ADR;                                              BRA01090
          MOCON = ' ';                                                    BRA01100
```

FILE: BRANCHG PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
        CMTS = 'HEX ADDRESS';                                   BRA01110
        CALL RITEOUT;                                           BRA01120
        GO TO A001;                                             BRA01130
                                                                BRA01140
END BRANCHG;                                                    BRA01150
```

```
FILE: NEOCG     PLIOPT    A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

 NEOCG : PROCEDURE:                                                          NE000010
                                                                             NE000020
     DCL  1    TMP_AD                  (70)      EXT.                       NE000030
               2    TMP_ADR            CHAR      (4):                       NE000040
     DCL  OP1_PTR                      FIXED     (4)   EXT.                 NE000050
          OP2_PTR                      FIXED     (4)   EXT.                 NE000060
          HEXCON                       CHAR      (2)   EXT.                 NE000070
          MOCON                        CHAR      (9)   EXT.                 NE000080
          CMTS                         CHAR      (45)  EXT:                 NE000090
     DCL  RITEOUT    ENTRY:                                                  NE000100
                                                                             NE000110
     /* THIS ROUTINE GENERATES CODE FOR CONDITIONAL BRANCHING */             NE000120
                                                                             NE000130
     /* COMPARE ARGUMENTS */                                                 NE000140
                                                                             NE000150
          HEXCON = '96':                                                     NE000160
          MOCON = 'LOAA':                                                    NE000170
          CMTS = 'LOAD FIRST ARGUMENT':                                      NE000180
          CALL RITEOUT:                                                      NE000190
                                                                             NE000200
          HEXCON = SUBSTR(TMP_ADR(OP1_PTR), 3, 2):                           NE000210
          MOCON = ' ':                                                       NE000220
          CMTS = 'INTO ACCUMULATOR A':                                       NE000230
          CALL RITEOUT:                                                      NE000240
                                                                             NE000250
          HEXCON = '90':                                                     NE000260
          MOCON = 'SUBA':                                                    NE000270
          CMTS = 'SUBTRACT SECOND ARGUMENT':                                 NE000280
          CALL RITEOUT:                                                      NE000290
                                                                             NE000300
          HEXCON = SUBSTR(TMP_ADR(OP2_PTR), 3, 2):                           NE000310
          MOCON = ' ':                                                       NE000320
          CMTS = 'FROM FIRST ARGUMENT':                                      NE000330
          CALL RITEOUT:                                                      NE000340
                                                                             NE000350
     /* COMPARE AND BRANCH */                                                NE000360
                                                                             NE000370
          HEXCON = '26':                                                     NE000380
          MOCON = 'BNE':                                                     NE000390
          CMTS = 'IF RESULT OF COMPARISON IS TRUE':                          NE000400
          CALL RITEOUT:                                                      NE000410
                                                                             NE000420
          HEXCON = '03':                                                     NE000430
          MOCON = ' ':                                                       NE000440
          CMTS = 'BRANCH TO NEXT STATEMENT':                                 NE000450
          CALL RITEOUT:                                                      NE000460
                                                                             NE000470
          RETURN:                                                            NE000480
                                                                             NE000490
 END NEOCG:                                                                  NE000500
```

```
FILE: CONEQCG  PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

 CONEQCG : PROCEDURE:                                                    CON00010
                                                                         CON00020
     DCL  1    TMP_AD                    (70)      EXT.                 CON00030
               2    TMP_ADR              CHAR      (4):                 CON00040
     DCL  OP1_PTR                        FIXED     (4)   EXT.           CON00050
          OP2_PTR                        FIXED     (4)   EXT.           CON00060
          HEXCON                         CHAR      (2)   EXT.           CON00070
          MOCON                          CHAR      (9)   EXT.           CON00080
          CMTS                           CHAR      (45)  EXT:           CON00090
     DCL  RITEOUT   ENTRY:                                               CON00100
                                                                         CON00110
     /* THIS ROUTINE GENERATES CODE FOR CONDITIONAL BRANCHING */         CON00120
                                                                         CON00130
     /* COMPARE ARGUMENTS */                                             CON00140
                                                                         CON00150
          HEXCON = '96':                                                 CON00160
          MOCON = 'LOAA':                                                CON00170
          CMTS = 'LOAD FIRST ARGUMENT':                                  CON00180
          CALL RITEOUT:                                                  CON00190
                                                                         CON00200
          HEXCON = SUBSTR(TMP_ADR(OP1_PTR), 3, 2):                       CON00210
          MOCON = ' ':                                                   CON00220
          CMTS = 'INTO ACCUMULATOR A':                                   CON00230
          CALL RITEOUT:                                                  CON00240
                                                                         CON00250
          HEXCON = '90':                                                 CON00260
          MOCON = 'SUBA':                                                CON00270
          CMTS = 'SUBTRACT SECOND ARGUMENT':                             CON00280
          CALL RITEOUT:                                                  CON00290
                                                                         CON00300
          HEXCON = SUBSTR(TMP_ADR(OP2_PTR), 3, 2):                       CON00310
          MOCON = ' ':                                                   CON00320
          CMTS = 'FROM FIRST ARGUMENT':                                  CON00330
          CALL RITEOUT:                                                  CON00340
                                                                         CON00350
     /* COMPARE AND BRANCH */                                            CON00360
                                                                         CON00370
          HEXCON = '27':                                                 CON00380
          MOCON = 'BEQ':                                                 CON00390
          CMTS = 'IF RESULT OF COMPARISON IS TRUE':                      CON00400
          CALL RITEOUT:                                                  CON00410
                                                                         CON00420
          HEXCON = '03':                                                 CON00430
          MOCON = ' ':                                                   CON00440
          CMTS = 'BRANCH TO NEXT STATEMENT':                             CON00450
          CALL RITEOUT:                                                  CON00460
                                                                         CON00470
          RETURN:                                                        CON00480
                                                                         CON00490
 END CONEQCG:                                                            CON00500
```

FILE: LTCG PLIOPT A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
LTCG : PROCEDURE:                                                    LTC00010
                                                                     LTC00020
   DCL  1   TMP_AD                    (70)    EXT,                   LTC00030
            2   TMP_ADR               CHAR    (4):                   LTC00040
   DCL  OP1_PTR                       FIXED   (4)  EXT,              LTC00050
        OP2_PTR                       FIXED   (4)  EXT,              LTC00060
        HEXCON                        CHAR    (2)  EXT,              LTC00070
        MOCON                         CHAR    (9)  EXT,              LTC00080
        CMTS                          CHAR    (45) EXT:              LTC00090
   DCL  RITEOUT   ENTRY:                                             LTC00100
                                                                     LTC00110
   /* THIS ROUTINE GENERATES CODE FOR CONDITIONAL BRANCHING */       LTC00120
                                                                     LTC00130
   /* COMPARE ARGUMENTS */                                           LTC00140
                                                                     LTC00150
        HEXCON = '96':                                               LTC00160
        MOCON = 'LOAA':                                              LTC00170
        CMTS = 'LOAD FIRST ARGUMENT':                                LTC00180
        CALL RITEOUT:                                                LTC00190
                                                                     LTC00200
        HEXCON = SUBSTR(TMP_ADR(OP1_PTR), 3, 2):                     LTC00210
        MOCON = ' ':                                                 LTC00220
        CMTS = 'INTO ACCUMULATOR A':                                 LTC00230
        CALL RITEOUT:                                                LTC00240
                                                                     LTC00250
        HEXCON = '90':                                               LTC00260
        MOCON = 'SUBA':                                              LTC00270
        CMTS = 'SUBTRACT SECOND ARGUMENT':                           LTC00280
        CALL RITEOUT:                                                LTC00290
                                                                     LTC00300
        HEXCON = SUBSTR(TMP_ADR(OP2_PTR), 3, 2):                     LTC00310
        MOCON = ' ':                                                 LTC00320
        CMTS = 'FROM FIRST ARGUMENT':                                LTC00330
        CALL RITEOUT:                                                LTC00340
                                                                     LTC00350
   /* COMPARE AND BRANCH */                                          LTC00360
                                                                     LTC00370
        HEXCON = '20':                                               LTC00380
        MOCON = 'BLT':                                               LTC00390
        CMTS = 'IF RESULT OF COMPARISON IS TRUE':                    LTC00400
        CALL RITEOUT:                                                LTC00410
                                                                     LTC00420
        HEXCON = '03':                                               LTC00430
        MOCON = ' ':                                                 LTC00440
        CMTS = 'BRANCH TO NEXT STATEMENT':                           LTC00450
        CALL RITEOUT:                                                LTC00460
                                                                     LTC00470
        RETURN:                                                      LTC00480
                                                                     LTC00490
END LTCG:                                                            LTC00500
```

```
FILE: GTCG      PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

 GTCG : PROCEDURE:                                                          GTC00010
                                                                            GTC00020
      DCL  1    TMP_AD                  (70)      EXT,                    GTC00030
           2      TMP_ADR               CHAR      (4):                    GTC00040
      DCL  OP1_PTR                     FIXED     (4)  EXT,                GTC00050
           OP2_PTR                     FIXED     (4)  EXT,                GTC00060
           HEXCON                      CHAR      (2)  EXT,                GTC00070
           MOCON                       CHAR      (9)  EXT,                GTC00080
           CMTS                        CHAR      (45) EXT;                GTC00090
      DCL  RITEOUT   ENTRY;                                                 GTC00100
                                                                            GTC00110
      /* THIS ROUTINE GENERATES CODE FOR CONDITIONAL BRANCHING */           GTC00120
                                                                            GTC00130
      /* COMPARE ARGUMENTS */                                               GTC00140
                                                                            GTC00150
           HEXCON = '96';                                                   GTC00160
           MOCON = 'LOAA';                                                  GTC00170
           CMTS = 'LOAD FIRST ARGUMENT';                                    GTC00180
           CALL RITEOUT;                                                    GTC00190
                                                                            GTC00200
           HEXCON = SUBSTR(TMP_ADR(OP1_PTR), 3, 2);                         GTC00210
           MOCON = ' ';                                                     GTC00220
           CMTS = 'INTO ACCUMULATOR A';                                     GTC00230
           CALL RITEOUT;                                                    GTC00240
                                                                            GTC00250
           HEXCON = '90';                                                   GTC00260
           MOCON = 'SUBA';                                                  GTC00270
           CMTS = 'SUBTRACT SECOND ARGUMENT';                               GTC00280
           CALL RITEOUT;                                                    GTC00290
                                                                            GTC00300
           HEXCON = SUBSTR(TMP_ADR(OP2_PTR), 3, 2);                         GTC00310
           MOCON = ' ';                                                     GTC00320
           CMTS = 'FROM FIRST ARGUMENT';                                    GTC00330
           CALL RITEOUT;                                                    GTC00340
                                                                            GTC00350
      /* COMPARE AND BRANCH */                                              GTC00360
                                                                            GTC00370
           HEXCON = '2E';                                                   GTC00380
           MOCON = 'BGT';                                                   GTC00390
           CMTS = 'IF RESULT OF COMPARISON IS TRUE';                        GTC00400
           CALL RITEOUT;                                                    GTC00410
                                                                            GTC00420
           HEXCON = '03';                                                   GTC00430
           MOCON = ' ';                                                     GTC00440
           CMTS = 'BRANCH TO NEXT STATEMENT';                               GTC00450
           CALL RITEOUT;                                                    GTC00460
                                                                            GTC00470
           RETURN;                                                          GTC00480
                                                                            GTC00490
 END GTCG;                                                                  GTC00500
```

```
FILE: GEOCG     PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

GEOCG : PROCEDURE:                                                          GE000010
                                                                            GE000020
    DCL  1    TMP_AD                (70)      EXT,                        GE000030
              2    TMP_ADR          CHAR      (4):                        GE000040
    DCL  OP1_PTR                    FIXED     (4)  EXT,                   GE000050
         OP2_PTR                    FIXED     (4)  EXT,                   GE000060
         HEXCON                     CHAR      (2)  EXT,                   GE000070
         MOCON                      CHAR      (9)  EXT,                   GE000080
         CMTS                       CHAR      (45) EXT:                   GE000090
    DCL  RITEOUT   ENTRY:                                                   GE000100
                                                                            GE000110
    /* THIS ROUTINE GENERATES CODE FOR CONDITIONAL BRANCHING */             GE000120
                                                                            GE000130
    /* COMPARE ARGUMENTS */                                                 GE000140
                                                                            GE000150
        HEXCON = '96':                                                      GE000160
        MOCON = 'LOAA':                                                     GE000170
        CMTS = 'LOAD FIRST ARGUMENT':                                       GE000180
        CALL RITEOUT:                                                       GE000190
                                                                            GE000200
        HEXCON = SUBSTR(TMP_ADR(OP1_PTR), 3, 2):                            GE000210
        MOCON = ' ':                                                        GE000220
        CMTS = 'INTO ACCUMULATOR A':                                        GE000230
        CALL RITEOUT:                                                       GE000240
                                                                            GE000250
        HEXCON = '90':                                                      GE000260
        MOCON = 'SUBA':                                                     GE000270
        CMTS = 'SUBTRACT SECOND ARGUMENT':                                  GE000280
        CALL RITEOUT:                                                       GE000290
                                                                            GE000300
        HEXCON = SUBSTR(TMP_ADR(OP2_PTR), 3, 2):                            GE000310
        MOCON = ' ':                                                        GE000320
        CMTS = 'FROM FIRST ARGUMENT':                                       GE000330
        CALL RITEOUT:                                                       GE000340
                                                                            GE000350
    /* COMPARE AND BRANCH */                                                GE000360
                                                                            GE000370
        HEXCON = '2C':                                                      GE000380
        MOCON = 'BGE':                                                      GE000390
        CMTS = 'IF RESULT OF COMPARISON IS TRUE':                           GE000400
        CALL RITEOUT:                                                       GE000410
                                                                            GE000420
        HEXCON = '03':                                                      GE000430
        MOCON = ' ':                                                        GE000440
        CMTS = 'BRANCH TO NEXT STATEMENT':                                  GE000450
        CALL RITEOUT:                                                       GE000460
                                                                            GE000470
        RETURN:                                                             GE000480
                                                                            GE000490
END GEOCG:                                                                  GE000500
```

```
FILE: LEQCG    PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

 LEQCG : PROCEDURE;                                                          LE000010
                                                                             LE000020
     DCL  1   TMP_AD                    (70)     EXT,                       LE000030
              2   TMP_ADR               CHAR     (4);                       LE000040
     DCL  1   LIT_AD                    (500)    EXT,                       LE000050
              2   LIT_ADR               CHAR     (4);                       LE000060
     DCL  1   IDN_AD                    (99)     EXT,                       LE000070
              2   IDN_ADR               CHAR     (4);                       LE000080
     DCL  OP1_PTR                       FIXED    (4)  EXT,                  LE000090
          OP2_PTR                       FIXED    (4)  EXT,                  LE000100
          OP1                           CHAR     (3)  EXT,                  LE000110
          OP2                           CHAR     (3)  EXT,                  LE000120
          HEXCON                        CHAR     (2)  EXT,                  LE000130
          MOCON                         CHAR     (9)  EXT,                  LE000140
          CMTS                          CHAR     (45) EXT;                  LE000150
     DCL  RITEOUT   ENTRY;                                                   LE000160
                                                                             LE000170
     /* THIS ROUTINE GENERATES CODE FOR CONDITIONAL BRANCHING */             LE000180
                                                                             LE000190
     /* COMPARE ARGUMENTS */                                                 LE000200
                                                                             LE000210
          HEXCON = '96';                                                     LE000220
          MOCON = 'LOAA';                                                    LE000230
          CMTS = 'LOAD FIRST ARGUMENT';                                      LE000240
          CALL RITEOUT;                                                      LE000250
                                                                             LE000260
          IF OP1 = 'TMP' THEN HEXCON = SUBSTR(TMP_ADR(OP1_PTR),3,2);         LE000270
          IF OP1 = 'IDN' THEN HEXCON = SUBSTR(IDN_ADR(OP1_PTR),3,2);         LE000280
          IF OP1 = 'LIT' THEN HEXCON = SUBSTR(LIT_ADR(OP1_PTR),3,2);         LE000290
          MOCON = ' ';                                                       LE000300
          CMTS = 'INTO ACCUMULATOR A';                                       LE000310
          CALL RITEOUT;                                                      LE000320
                                                                             LE000330
          HEXCON = '90';                                                     LE000340
          MOCON = 'SUBA';                                                    LE000350
          CMTS = 'SUBTRACT SECOND ARGUMENT ';                                LE000360
          CALL RITEOUT;                                                      LE000370
                                                                             LE000380
          IF OP2 = 'TMP' THEN HEXCON = SUBSTR(TMP_ADR(OP2_PTR),3,2);         LE000390
          IF OP2 = 'IDN' THEN HEXCON = SUBSTR(IDN_ADR(OP2_PTR),3,2);         LE000400
          IF OP2 = 'LIT' THEN HEXCON = SUBSTR(LIT_ADR(OP2_PTR),3,2);         LE000410
          MOCON = ' ';                                                       LE000420
          CMTS = 'FROM FIRST ARGUMENT';                                      LE000430
          CALL RITEOUT;                                                      LE000440
                                                                             LE000450
     /* COMPARE AND BRANCH */                                                LE000460
                                                                             LE000470
          HEXCON = '2F';                                                     LE000480
          MOCON = 'BLE';                                                     LE000490
          CMTS = 'IF RESULT OF COMPARISON IS TRUE';                          LE000500
          CALL RITEOUT;                                                      LE000510
                                                                             LE000520
          HEXCON = '03';                                                     LE000530
          MOCON = ' ';                                                       LE000540
          CMTS = 'BRANCH TO NEXT STATEMENT';                                 LE000550
```

```
FILE: LEOCG    PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

        CALL RITEOUT:                                                    LE000560
                                                                         LE000570
        RETURN:                                                          LE000580
                                                                         LE000590
END LEOCG:                                                               LE000600
```

```
FILE: HXDCCON  PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

 HXDCCON : PROCEDURE:                                                          HXD00010
                                                                               HXD00020
      DCL  1    LAL_AD                    (500)     EXT,                      HXD00030
           2    LAL_PTR                   FIXED     (4),                      HXD00040
           2    LAL_ADR                   CHAR      (4);                      HXD00050
      DCL  DECDEST                        FIXED     (5)   EXT,               HXD00060
           L                              FIXED     (1),                      HXD00070
           A                              CHAR      (1),                      HXD00080
           X                              FIXED     (2),                      HXD00090
           R                              FIXED     (5),                      HXD00100
           FACTOR                         FIXED     (2),                      HXD00110
           LAL_IDX                        FIXED     (4)   EXT;                HXD00120
                                                                               HXD00130
     /* THIS ROUTINE CONVERTS A HEX CHARACTER FIELD TO A DECIMAL */            HXD00140
                                                                               HXD00150
           DECDEST = 0;                                                        HXD00160
           DO L = 0 TO 3 BY 1;                                                 HXD00170
                A = SUBSTR(LAL_ADR(LAL_IDX), 4-L, 1);                          HXD00180
                X = VERIFY(A, '0123456789');                                   HXD00190
                IF X = 0 THEN FACTOR = A;                                      HXD00200
                IF A = 'A' THEN FACTOR = 10;                                   HXD00210
                IF A = 'B' THEN FACTOR = 11;                                   HXD00220
                IF A = 'C' THEN FACTOR = 12;                                   HXD00230
                IF A = 'D' THEN FACTOR = 13;                                   HXD00240
                IF A = 'E' THEN FACTOR = 14;                                   HXD00250
                IF A = 'F' THEN FACTOR = 15;                                   HXD00260
                R = FACTOR * (16 ** L);                                        HXD00270
                DECDEST = DECDEST + R;                                         HXD00280
           END;                                                                HXD00290
           RETURN;                                                             HXD00300
                                                                               HXD00310
 END HXDCCON;                                                                  HXD00320
```

```
FILE: HEADING  PLIOPT   A              YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

 HEADING : PROCEDURE;                                                    HEA00010
                                                                         HEA00020
      DCL DATAOUT   STREAM    PRINT;                                     HEA00030
                                                                         HEA00040
          PUT FILE (DATAOUT) PAGE;                                       HEA00050
          PUT FILE (DATAOUT) SKIP EDIT ('MNEMONICS/') (X(25), A(10));    HEA00060
          PUT FILE (DATAOUT) SKIP EDIT ('HEX', 'HEX', 'DECIMAL')         HEA00070
               (X(5), A(3), X(7), A(3), X(7), A(7));                     HEA00080
          PUT FILE (DATAOUT) SKIP EDIT ('ADDRESS', 'CONTENTS',           HEA00090
               'CONTENTS', 'COMMENTS') (X(5), A(7), X(3), A(8), X(2),    HEA00100
               A(8), X(7), A(8));                                        HEA00110
          PUT FILE (DATAOUT) SKIP;                                       HEA00120
          RETURN;                                                        HEA00130
 END HEADING;                                                            HEA00140
```

```
FILE: DCHXCD   PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

DCHXCD : PROCEDURE;                                                          DCH00010
                                                                             DCH00020
   DCL  HEXADR                  CHAR      (4)  EXT,                         DCH00030
        H (4)                   CHAR      (1),                              DCH00040
        N                       FIXED     (1),                              DCH00050
        A                       FIXED     (5),                              DCH00060
        B                       FIXED     (5),                              DCH00070
        C                       FIXED     (5),                              DCH00080
        D                       FIXED     (2),                              DCH00090
        DECADR                  FIXED     (5)  EXT;                         DCH00100
   DCL  DATAOUT   STREAM        PRINT;                                       DCH00110
                                                                             DCH00120
    /* INITIALIZE*/                                                         DCH00130
                                                                             DCH00140
        DO N=1 TO 4 BY 1;                                                    DCH00150
           H(N)='0';                                                         DCH00160
        END;                                                                 DCH00170
        N=1;                                                                 DCH00180
                                                                             DCH00190
    /* BEGIN CONVERSION */                                                  DCH00200
                                                                             DCH00210
        A = DECADR;                                                          DCH00220
DH001:                                                                       DCH00230
        B=A/16;                                                              DCH00240
        C=TRUNC(B);                                                          DCH00250
        D=(MOD(A,16))+1;                                                     DCH00260
        H(N) = SUBSTR ('0123456789ABCDEF', D, 1);                            DCH00270
        IF A < 16 THEN DO;                                                   DCH00280
           HEXADR = H(4) || H(3) || H(2) || H(1);                            DCH00290
           RETURN;                                                           DCH00300
        END;                                                                 DCH00310
        A = C;                                                               DCH00320
        N = N + 1;                                                           DCH00330
        GO TO DH001;                                                         DCH00340
END DCHXCD;                                                                  DCH00350
```

```
FILE: DCHXCD2  PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

 DCHXCD2 : PROCEDURE:                                                          DCH00010
                                                                               DCH00020
     DCL  HEXCON                    CHAR      (2)  EXT,                      DCH00030
          MDCON                     CHAR      (9)  EXT,                      DCH00040
          H (2)                     CHAR      (1),                           DCH00050
          N                         FIXED     (1),                           DCH00060
          A                         FIXED     (5),                           DCH00070
          B                         FIXED     (5),                           DCH00080
          C                         FIXED     (5),                           DCH00090
          D                         FIXED     (2),                           DCH00100
          F                         CHAR      (9)  INIT((9)'0');             DCH00110
     DCL  DATAOUT   STREAM          PRINT;                                     DCH00120
                                                                               DCH00130
     /* INITIALIZE */                                                         DCH00140
                                                                               DCH00150
          H(1)='0';                                                            DCH00160
          H(2)='0';                                                            DCH00170
                                                                               DCH00180
     /* BEGIN CONVERSION */                                                   DCH00190
                                                                               DCH00200
          A = INDEX (MDCON, ' ');                                              DCH00210
          MDCON = SUBSTR (F, 1, 10-A) || SUBSTR (MDCON, 1, A-1);               DCH00220
          N = 1;                                                               DCH00230
          A = MDCON;                                                           DCH00240
DH2001:                                                                        DCH00250
          B = A/16;                                                            DCH00260
          C = TRUNC (B);                                                       DCH00270
          D = (MOD(A,16)) + 1;                                                 DCH00280
          H(N) = SUBSTR ('0123456789ABCDEF', D, 1);                            DCH00290
          IF A < 16 THEN                                                       DCH00300
               DO;                                                             DCH00310
                    HEXCON = H(2) || H(1);                                     DCH00320
                    RETURN;                                                    DCH00330
               END;                                                            DCH00340
          A = C;                                                               DCH00350
          N = N + 1;                                                           DCH00360
          GO TO DH2001;                                                        DCH00370
END DCHXCD2;                                                                   DCH00380
```

```
FILE: RITEOUT  PLIOPT   A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

 RITEOUT : PROCEDURE:                                                          RIT00010
                                                                               RIT00020
      DCL  HEXADR                    CHAR      (4)  EXT.                     RIT00030
           HEXCON                    CHAR      (2)  EXT.                     RIT00040
           MDCON                     CHAR      (9)  EXT.                     RIT00050
           CMTS                      CHAR      (45) EXT.                     RIT00060
           PGCNT                     FIXED     (2)  EXT.                     RIT00070
           DECADR                    FIXED     (5)  EXT:                     RIT00080
      DCL  DCHXCO                    ENTRY:                                  RIT00090
      DCL  HEADING                   ENTRY:                                  RIT00100
      DCL  DATAOUT     STREAM        PRINT:                                  RIT00110
                                                                               RIT00120
      /* CONVERT DECIMAL ADDRESS TO HEXADECIMAL ADDRESS */                     RIT00130
                                                                               RIT00140
           CALL DCHXCO:                                                        RIT00150
                                                                               RIT00160
      /* WRITE TO OUTPUT FILE */                                               RIT00170
                                                                               RIT00180
           PUT FILE (DATAOUT) SKIP EDIT (HEXADR, HEXCON, MDCON, CMTS)          RIT00190
                (X(5), A(4), X(6), A(2), X(8), A(10), X(6), A(45)):            RIT00200
                                                                               RIT00210
      /* UPDATE MEMORY ADDRESS AND OUTPUT PAGE LINE COUNTER */                 RIT00220
                                                                               RIT00230
           PGCNT = PGCNT + 1:                                                  RIT00240
           DECADR = DECADR + 1:                                                RIT00250
           IF PGCNT > 49 THEN                                                  RIT00260
                DO:                                                            RIT00270
                     CALL HEADING:                                             RIT00280
                     PGCNT = 0:                                                RIT00290
                END:                                                           RIT00300
           RETURN:                                                             RIT00310
 END RITEOUT:                                                                  RIT00320
```

APPENDIX C

Sample Programs

HEX ADDRESS	HEX CONTENTS	MNEMONICS/ DECIMAL CONTENTS	COMMENTS
0000	02	000000002	DATA
0001	04	000000004	DATA
0002	06	000000006	DATA
0003	01	000000001	CONSTANT
0004	63	000000099	CONSTANT
0005	02	000000002	CONSTANT
0006	03	000000003	CONSTANT
0007	04	000000004	CONSTANT
0008	06	000000006	CONSTANT
0009	05	000000005	CONSTANT
000A	A4	000004004	CONSTANT
000B	07	000000007	CONSTANT
000C	00	000000013	CONSTANT
000D	08	000000008	CONSTANT
000E	0F	000000015	CONSTANT
000F	09	000000009	CONSTANT
0010	A6	000004006	CONSTANT
0011	0A	000000010	CONSTANT
0012	0B	000000011	CONSTANT
0013	0C	000000012	CONSTANT
0014	0E	000000014	CONSTANT
0015	10	000000016	CONSTANT
0016	11	000000017	CONSTANT
0017	XX	XXXXXXXXX	K
0018	XX	XXXXXXXXX	T
0019	XX	XXXXXXXXX	A
001A	XX	XXXXXXXXX	B
001B	XX	XXXXXXXXX	C
001C	XX	XXXXXXXXX	X -
001D	XX	XXXXXXXXX	TEMPORARY STORAGE LOCATION
001E	XX	XXXXXXXXX	TEMPORARY STORAGE LOCATION
001F	XX	XXXXXXXXX	TEMPORARY STORAGE LOCATION
0020	XX	XXXXXXXXX	TEMPORARY STORAGE LOCATION
0021	XX	XXXXXXXXX	TEMPORARY STORAGE LOCATION
0100	CE	LDX	RESET INDEX REGISTER TO
0101	00		TOP OF DATA TABLE STARTING
0102	00		AT ADDRESS 0000
0103	96	LDAA	LOAD ACCUMULATOR A DIRECT
0104	03		WITH CONSTANT AT THIS ADDRESS
0105	97	STAA	STORE ACCUMULATOR A DIRECT
0106	17		IN VARIABLE NAME AT THIS ADDRESS
0107	96	LDAA	LOAD FIRST ARGUMENT
0108	17		INTO ACCUMULATOR A
0109	90	SUBA	SUBTRACT SECOND ARGUMENT
010A	08		FROM FIRST ARGUMENT
010B	2F	BLE	IF RESULT OF COMPARISON IS TRUE
010C	03		BRANCH TO NEXT STATEMENT
010D	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
010E	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
010F	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER

HEX ADDRESS	HEX CONTENTS	MNEMONICS/ DECIMAL CONTENTS	COMMENTS
0110	B6	LDAA	LOAD DATA FROM INPUT
0111	40		DEVICE AT THIS ADDRESS
0112	04		INTO ACCUMULATOR A, THEN
0113	97	STAA	STORE DATA IN VARIABLE
0114	18		NAME AT THIS ADDRESS
0115	A6	LDAA, X	LOAD DATA INTO ACCUMULATOR A USING
0116	00		INDEXED ADDRESSING WITH ZERO OFFSET
0117	97	STAA	STORE DATA DIRECT INTO VARIABLE NAME
0118	19		AT THIS ZERO PAGE ADDRESS
0119	08	INX	INCREMENT INDEX REGISTER
011A	A6	LDAA, X	LOAD DATA INTO ACCUMULATOR A USING
011B	00		INDEXED ADDRESSING WITH ZERO OFFSET
011C	97	STAA	STORE DATA DIRECT INTO VARIABLE NAME
011D	1A		AT THIS ZERO PAGE ADDRESS
011E	08	INX	INCREMENT INDEX REGISTER
011F	A6	LDAA, X	LOAD DATA INTO ACCUMULATOR A USING
0120	00		INDEXED ADDRESSING WITH ZERO OFFSET
0121	97	STAA	STORE DATA DIRECT INTO VARIABLE NAME
0122	1B		AT THIS ZERO PAGE ADDRESS
0123	08	INX	INCREMENT INDEX REGISTER
0124	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0125	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0126	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0127	96	LDAA	LOAD ACCUMULATOR A DIRECT
0128	17		WITH IDENTIFIER VALUE AT THIS ADDRESS
0129	97	STAA	STORE ACCUMULATOR A DIRECT
012A	1D		IN TEMPORARY STORAGE AT THIS ADDRESS
012B	96	LDAA	LOAD ACCUMULATOR A DIRECT
012C	09		WITH CONSTANT AT THIS ADDRESS
012D	97	STAA	STORE ACCUMULATOR A DIRECT
012E	1F		IN TEMPORARY STORAGE AT THIS ADDRESS
012F	96	LDAA	LOAD FIRST ARGUMENT
0130	1D		INTO ACCUMULATOR A
0131	90	SUBA	SUBTRACT SECOND ARGUMENT
0132	1E		FROM FIRST ARGUMENT
0133	27	BEQ	IF RESULT OF COMPARISON IS TRUE
0134	03		BRANCH TO NEXT STATEMENT
0135	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0136	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0137	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0138	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0139	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
013A	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
013B	96	LDAA	LOAD DATA STORED IN VARIABLE
013C	1C		AT THIS ADDRESS INTO ACCUMULATOR A,
013D	B7	STAA	THEN WRITE IT TO
013E	40		THE OUTPUT DEVICE RESIDING
013F	06		AT THIS HEX ADDRESS
0140	CE	LDX	RESET INDEX REGISTER TO
0141	00		TOP OF DATA TABLE STARTING

HEX ADDRESS	HEX CONTENTS	MNEMONICS/ DECIMAL CONTENTS	COMMENTS
0142	00		AT ADDRESS 0000
0143	96	LDAA	LOAD ACCUMULATOR A DIRECT
0144	17		WITH FIRST OPERAND
0145	9B	ADDA	ADD DIRECT TO ACCUMULATOR A
0146	03		THE SECOND OPERAND
0147	97	STAA	STORE ACCUMULATOR A DIRECT
0148	1D		IN THIS TEMPORARY LOCATION
0149	96	LDAA	LOAD ACCUMULATOR A DIRECT
014A	1D		WITH TEMP VALUE AT THIS ADDRESS
014B	97	STAA	STORE ACCUMULATOR A DIRECT
014C	17		IN VARIABLE NAME AT THIS ADDRESS
014D	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
014E	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
014F	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0150	3F	WAI	HALT EXECUTION (WAIT FOR INTERRUPT)
0151	96	LDAA	LOAD ACCUMULATOR A DIRECT
0152	19		WITH FIRST OPERAND
0153	9B	ADDA	ADD DIRECT TO ACCUMULATOR A
0154	1A		THE SECOND OPERAND
0155	97	STAA	STORE ACCUMULATOR A DIRECT
0156	1D		IN THIS TEMPORARY LOCATION
0157	7F	CLR	INITIALIZE
0158	00		ANSWER
0159	1F		AT ZERO
015A	20	BRA	CREATE A SCRATCHPAD
015B	01		LOCATION FOR
015C	XX		SQUARE OF ANSWER
015D	D6	LDAB	LOAD THE B ACCUMULATOR WITH
015E	1E		THE MULTIPLIER FROM THIS LOCATION
015F	4F	CLRA	CLEAR ACCUMULATOR A
0160	B7	STAA	STORE THE PRODUCT IN
0161	01		THIS SCRATCHPAD
0162	5C		LOCATION
0163	17	TBA	TRANSFER ACCUMULATOR B TO A
0164	27	BEQ	IF THE MULTIPLIER IS ZERO,
0165	09		CONTINUE WITH ALGORITHM
0166	4A	DECA	OTHERWISE, DECREMENT MULTIPLIER
0167	16	TAB	TRANSFER ACCUMULATOR A TO B
0168	B6	LDAA	LOAD THE ACCUMULATOR WITH
0169	01		THE PRODUCT STORED IN
016A	5C		THIS SCRATCHPAD LOCATION
016B	9B	ADDA	ADD TO THE PRODUCT THE
016C	1E		MULTIPLICAND AT THIS ADDRESS
016D	20	BRA	BRANCH BACK TO BEGINNING
016E	F1		OF MULTIPLY SEQUENCE
016F	B6	LDAA	LOAD ANSWER
0170	01		SQUARED INTO
0171	5C		ACCUMULATOR A
0172	90	SUBA	SUBTRACT THE ORIGINAL
0173	1D		NUMBER FROM ACCUMULATOR A

HEX ADDRESS	HEX CONTENTS	MNEMONICS/ DECIMAL CONTENTS	COMMENTS
0174	2C	BGE	IF ANSWER SQUARED IS GREATER THAN
0175	05		OR EQUAL TO ARGUMENT, GO TO NEXT ALGORITHM
0176	7C	INC	OTHERWISE, INCREMENT THE
0177	00		ANSWER AND
0178	1F		RETURN TO
0179	20	BRA	SQUARING
017A	E2		ROUTINE
017B	96	LDAA	LOAD ACCUMULATOR A DIRECT
017C	17		WITH FIRST OPERAND
017D	90	SUBA	SUBTRACT DIRECT FROM ACCUMULATOR A
017E	18		THE SECOND OPERAND
017F	97	STAA	STORE ACCUMULATOR A DIRECT
0180	1F		IN THIS TEMPORARY LOCATION
0181	96	LDAA	LOAD ACCUMULATOR A DIRECT
0182	1E		WITH FIRST OPERAND
0183	9B	ADDA	ADD DIRECT TO ACCUMULATOR A
0184	1F		THE SECOND OPERAND
0185	97	STAA	STORE ACCUMULATOR A DIRECT
0186	20		IN THIS TEMPORARY LOCATION
0187	D6	LDAB	LOAD THE B ACCUMULATOR WITH THE
0188	20		MULTIPLIER FROM THIS LOCATION
0189	4F	CLRA	CLEAR ACCUMULATOR A
018A	97	STAA	STORE THE PRODUCT IN
018B	21		THIS TEMPORARY LOCATION
018C	17	TBA	TRANSFER ACCUMULATOR B TO A
018D	27	BEQ	IF THE MULTIPLIER IS ZERO, BRANCH
018E	08		TO THE NEXT ALGORITHM
018F	4A	DECA	OTHERWISE, DECREMENT THE MULTIPLIER
0190	16	TAB	TRANSFER ACCUMULATOR A TO B
0191	96	LDAA	LOAD THE ACCUMULATOR WITH THE
0192	21		PRODUCT STORED IN THIS TEMPORARY LOCATION
0193	9B	ADDA	ADD TO THE PRODUCT
0194	05		MULTIPLICAND AT THIS ADDRESS
0195	20	BRA	BRANCH BACK TO BEGINNING OF
0196	F3		MULTIPLY ALGORITHM
0197	96	LDAA	LOAD ACCUMULATOR A DIRECT
0198	21		WITH TEMP VALUE AT THIS ADDRESS
0199	97	STAA	STORE ACCUMULATOR A DIRECT
019A	1C		IN VARIABLE NAME AT THIS ADDRESS
019B	39	RTS	RETURN FROM SUBROUTINE
019C	20	BRA	CREATE A SCRATCHPAD
019D	01		LOCATION FOR
019E	XX		EXPONENT
019F	86	LDAA	SET RESULT OF
01A0	01		EXPONENTIATION
01A1	97	STAA	TO
01A2	10		ONE
01A3	96	LDAA	LOAD VALUE OF
01A4	19		EXPONENT
01A5	87	STAA	INTO

HEX ADDRESS	HEX CONTENTS	MNEMONICS/ DECIMAL CONTENTS	COMMENTS
0186	01		SCRATCHPAD
0187	9F		LOCATION
0188	27	BEQ	IF EXPONENT IS ZERO
0189	IF		GO ON TO NEXT ALGORITHM
018A	20	BRA	CREATE A SCRATCHPAD
018B	01		LOCATION FOR
018C	XX		PRODUCT
018D	D6	LDAB	LOAD ACCUMULATOR B WITH THE
018F	1A		MULTIPLIER FROM THIS LOCATION
018F	4F	CLRA	CLEAR ACCUMULATOR A
0190	B7	STAA	STORE THE PRODUCT IN
0191	01		THIS
0192	AC		LOCATION
0193	17	TBA	TRANSFER ACCUMULATOR B TO A
0194	27	BEQ	IF THE MULTIPLIER IS ZERO, EXIT
0195	09		THE MULTIPLIER LOOP
0196	4A	DECA	OTHERWISE, DECREMENT THE MULTIPLIER
0197	16	TAB	TRANSFER ACCUMULATOR A TO B
0198	B6	LDAA	LOAD ACCUMULATOR
0199	01		WITH PRODUCT STORED
019A	AC		IN THIS LOCATION
019B	9B	ADDA	ADD TO THE PRODUCT THE
019C	1D		MULTIPLICAND AT THIS ADDRESS
019D	20	BRA	BRANCH BACK TO BEGINNING
019E	F1		OF MULTIPLY ALGORITHM
01BF	B6	LDAA	LOAD ACCUMULATOR A
01C0	01		WITH PRODUCT STORED
01C1	AC		IN THIS LOCATION
01C2	97	STAA	STORE RESULT IN THIS
01C3	1D		TEMPORARY STORAGE LOCATION
01C4	7A	DEC	DECREMENT
01C5	01		EXPONENT
01C6	9F		LOCATION
01C7	20	BRA	BRANCH BACK TO BEGINNING
01C8	DF		OF EXPONENTIATION ALGORITHM
01C9	D6	LDAB	LOAD THE B ACCUMULATOR WITH THE
01CA	1D		DIVIDEND FROM THIS LOCATION
01CB	4F	CLRA	CLEAR ACCUMULATOR A
01CC	97	STAA	STORE THE QUOTIENT IN
01CD	1F		THIS TEMPORARY LOCATION
01CE	17	TBA	TRANSFER ACCUMULATOR B TO A
01CF	90	SUBA	SUBTRACT FROM THE DIVIDEND THE
01D0	05		DIVISOR AT THIS ADDRESS
01D1	2B	BMI	IF THE DIFFERENCE IS NEGATIVE,
01D2	06		BRANCH TO THE NEXT ALGORITHM
01D3	16	TAB	OTHERWISE, TRANSFER ACCUMULATOR A TO B
01D4	96	LDAA	LOAD THE ACCUMULATOR WITH THE
01D5	1F		QUOTIENT STORED IN THIS TEMPORARY LOCATION
01D6	4C	INCA	INCREMENT THE QUOTIENT
01D7	20	BRA	BRANCH BACK TO BEGINNING OF

HEX ADDRESS	HEX CONTENTS	MNEMONICS/ DECIMAL CONTENTS	COMMENTS
0108	F3		DIVIDE ALGORITHM
0109	96	LDAA	LOAD ACCUMULATOR A DIRECT
010A	1F		WITH TEMP VALUE AT THIS ADDRESS
010B	97	STAA	STORE ACCUMULATOR A DIRECT
010C	1C		IN VARIABLE NAME AT THIS ADDRESS
010D	39	RTS	RETURN FROM SUBROUTINE
010E	3E	WAI	HALT EXECUTION (WAIT FOR INTERRUPT)
010D	7E	JMP	JUMP TO
010E	01		THIS
010F	50		HEX ADDRESS
0124	BD	JSR	JUMP TO SUBROUTINE
0125	01		BEGINNING AT THIS
012B	51		MEMORY LOCATION
0135	7E	JMP	JUMP TO
0136	01		THIS
0137	3B		HEX ADDRESS
0138	7E	JMP	JUMP TO
0139	01		THIS
013A	9C		HEX ADDRESS
0140	7E	JMP	JUMP TO
014E	01		THIS
014F	07		HEX ADDRESS

PROGRAM STARTING ADDRESS 0100

TOTAL MEMORY USED : 479 BYTES

SOURCE LISTING

```
1     REM THIS PROGRAM DEMONSTRATES COMPILER OPERATION.                    1
2     REM NO SOURCE PROGRAM ERRORS RESULTS IN THE GENERATION OF HEX CODE.  2
3     DATA 2.4.6                                                           3
4     FOR K=1 TO 6                                                         4
5     READ T FROM 4004                                                     5
6     READ A,B,C                                                           6
7     GOSUB 13                                                             7
8     IF K=5 GOTO 15                                                       8
9     WRITE X TO 4006                                                      9
10    RESTORE                                                              10
11    NEXT K                                                               11
12    STOP                                                                 12
13    LET X = (SQR(A+B)+K-T) * 2                                           13
14    RETURN                                                               14
15    LET X = (AAA)/2                                                      15
16    RETURN                                                               16
17    END                                                                  17
```

ERROR TABLE

ERROR	SOURCE LINE
NO DIAGNOSTICS GENERATED	0
	0
MAXIMUM TEMPORARY STORAGE USED -	5

KEYWORD TABLE

LET	1
GOTO	2
IF	3
TO	4
THEN	5
FOR	6
NEXT	7
GOSUB	8
RETURN	9
STEP	10
REM	11
DATA	12
READ	13
RESTORE	14
FROM	15
STOP	16
END	17
WRTOUT	18
ROIN	19

TERMINAL TABLE

-	1
+	2
/	3
*	4
↑	5
=	6
(	7
)	8
>	9
<	10
,	11
.	12
ABS	13
ATN	14
COS	15
DEF	16
EXP	17
INT	18
LOG	19
SIN	20
SQR	21
TAN	22
$	23
LEQ	24
GEQ	25
NEQ	26
CEQ	27

IDENTIFIER TABLE

K	1
T	2
A	3
B	4
C	5
X	6

LITERAL TABLE

1	1
99	2
2	3
3	4
4	5
6	6
5	7
4004	8
7	9
13	10
9	11
15	12
9	13
4006	14
10	15
11	16
12	17
14	18
16	19
17	20

UNIFORM SYMBOL TABLE

SYMBOL	POINTER
LIT	1
TRM	23
LIT	3
TRM	23
LIT	4
KEY	12
LIT	3
TRM	11
LIT	5
TRM	11
LIT	6
TRM	23
LIT	5
KEY	6
IDN	1
TRM	6
LIT	1
KEY	4
LIT	6
TRM	23
LIT	7
KEY	19
IDN	2
KEY	15
LIT	9
TRM	23
LIT	6
KEY	13
IDN	3
TRM	11
IDN	4
TRM	11
IDN	5
TRM	23
LIT	9
KEY	8
LIT	10
TRM	23
LIT	11
KEY	3
IDN	1
TRM	6
LIT	7
KEY	2
LIT	12
TRM	23
LIT	13
KEY	18
IDN	6
KEY	6
LIT	14
TRM	23
LIT	15
KEY	14
TRM	23
LIT	16

KEY 7
IDN 1
TRM 23
LIT 17
KEY 16
TRM 23
LIT 10
KEY 1
IDN 6
TRM 6
TRM 7
TRM 21
TRM 7
IDN 3
TRM 2
IDN 4
TRM 8
TRM 2
IDN 1
TRM 1
IDN 2
TRM 8
TRM 4
LIT 3
TRM 23
LIT 18
KEY 9
TRM 23
LIT 12
KEY 1
IDN 6
TRM 6
TRM 7
IDN 4
TRM 5
IDN 3
TRM 8
TRM 3
LIT 3
TRM 23
LIT 19
KEY 9
TRM 23
LIT 20
KEY 17
TRM 23

DATA TABLE

LIT TBL POINTER
3
5
6

SOURCE LISTING

```
1     REM THIS PROGRAM PRODUCES GENERAL ERROR MESSAGES.                    1
2     LET A = 12B0                                                         2
3     COMPUTE A+B                                                          3
40A   LET A = 12                                                           4
 5    LET A = 12                                                           5
123456 LET A=12                                                           6
7                                                          LET A=         7
8     STOP                                                                 8
```

ERROR TABLE

ERROR	SOURCE LINE
NUMERIC NOT FOUND IN COLUMN ONE	5
LINE NUMBER EXCEEDS FIVE DIGITS	6
CHARACTERS FOUND BEYOND COLUMN 72	7
ILLEGAL CHARACTER GROUP	2
ILLEGAL CHARACTER GROUP	3
ILLEGAL CHARACTER GROUP	4
ILLEGAL KEYWORD FOLLOWS LINE NUMBER	3
LINE DOES NOT START WITH A VALID NUMBER	4
LAST PROGRAM LINE MUST BE END STMT	9
	0
ABEND - SEVERE ERRORS DETECTED.	0
CODE GENERATION SUPPRESSED.	0

SOURCE LISTING

```
1     REM THIS PROGRAM PRODUCES  ERROR MESSAGES FOR LET, RDIN,          1
2     REM AND IF STATEMENTS.                                           2
3     LET A13 = 5                                                      3
4     LET A1 5 =                                                       4
5     LET X = SQR 2                                                    5
6     RDIN 15                                                          6
7     RDIN S .T                                                        7
8     RDIN S FROM P                                                    8
9     RDIN S FROM 4004 , 4005                                          9
10    IF A + B THEN 5                                                  10
11    IF A=B THEN B=X                                                  11
12    IF A=B THEN A+B                                                  12
13    IF (A+B)-3) = 7 THEN 5                                           13
14    STOP                                                             14
15    END                                                              15
```

ERROR TABLE

ERROR	SOURCE LINE
ILLEGAL CHARACTER GROUP	3
IDENTIFIER MUST FOLLOW LET	3
EQUAL SIGN IS NOT IN PROPER POSITION	4
PARENTHESIS MUST ENCLOSE FUNCTION ARG	5
VARIABLE NAME MUST FOLLOW RDIN	6
FROM MUST FOLLOW VARIABLE NAME	7
HEX ADDRESS MUST FOLLOW FROM	8
MULTIPLE ENTRIES FOR HEX ADDRESS	9
REL. OP. MUST FOLLOW FIRST CONDITION	10
INVALID COMMAND FOLLOWS CONDITION	11
INVALID COMMAND FOLLOWS CONDITION	12
UNEQUAL NO. OF LEFT AND RIGHT PAREN.	13
REL. OP. MUST FOLLOW FIRST CONDITION	14
	0
	0
ABEND = SEVERE ERRORS DETECTED.	0
CODE GENERATION SUPPRESSED.	0

SOURCE LISTING

```
1     REM THIS PROGRAM PRODUCES ERROR MESSAGES FOR DATA, RETURN, STOP,      1
2     REM, WRTOUT, GOTO, RESTORE, READ AND END STATEMENTS                   2
3     DATA A,B,C                                                            3
4     DATA 1 2 3                                                            4
5     RETURN TO A                                                           5
6     STOP EXECUTION                                                        6
7     WRTOUT 15                                                             7
8     WRTOUT A INTO 4004                                                    8
9     WRTOUT A TO B                                                         9
10    WRTOUT A TO 9006 .9007                                               10
11    GOTO A                                                               11
12    GOTO 1 2                                                             12
13    RESTORE A                                                            13
14    READ A12                                                             14
15    READ A B C                                                           15
16    END EXECUTION                                                        16
17    END                                                                  17
```

ERROR TABLE

ERROR	SOURCE LINE
ILLEGAL CHARACTER GROUP	6
ILLEGAL CHARACTER GROUP	8
ILLEGAL CHARACTER GROUP	14
ILLEGAL CHARACTER GROUP	16
DATA ENTRIES MUST BE NUMERIC	3
COMMAS REQUIRED BETWEEN DATA VALUES	4
CHARACTERS APPEAR AFTER RETURN STMT	5
VARIABLE NAME MUST FOLLOW WTOUT	7
TO MUST FOLLOW VARIABLE NAME	8
HEX ADDRESS MUST FOLLOW TO	9
MULTIPLE ENTRIES FOR HEX ADDRESS	10
LINE NUMBER MUST FOLLOW GOTO STMT	11
MULTIPLE ARGUMENTS IN GOTO STMT	12
CHARACTERS APPEAR AFTER RESTORE STMT	13
READ ARGUMENT MUST BE A VARIABLE NAME	14
COMMAS REQUIRED BETWEEN READ ARGUMENTS	15
	0
	0
ABEND - SEVERE ERRORS DETECTED.	0
CODE GENERATION SUPPRESSED.	0

SOURCE LISTING

```
1     REM THIS PROGRAM PRODUCES ERROR MESSAGES FOR NEXT, GOSUB                1
2     REM AND FOR STATEMENTS.                                                 2
3     FOR 12 TO A                                                             3
4     FOR L 1 = TO 7                                                          4
5     FOR L=1.5 TO 2.5                                                        5
6     FOR A57 = 1 TO 7                                                        6
7     FOR A = 1  7                                                            7
8     FOR A = 1 TO 7                                                          8
9     FOR A = 1 TO A57                                                        9
10    FOR A = 1 TO 2.5                                                        10
11    FOR A = 1 TO 7 LET K = 1                                                11
12    FOR A = 1 TO 7 STEP A57                                                 12
13    FOR A = 1 TO 7 STEP 1.5                                                 13
14    FOR A = 1 TO 7 X                                                        14
15    NEXT 12                                                                 15
16    NEXT Q                                                                  16
17    NEXT L 1                                                                17
18    GOSUB A                                                                 18
19    GOSUB 1 2                                                               19
20    NEXT A                                                                  20
21    END                                                                     21
```

ERROR TABLE

ERROR	SOURCE LINE
ILLEGAL CHARACTER GROUP	6
ILLEGAL CHARACTER GROUP	9
ILLEGAL CHARACTER GROUP	12
IDENTIFIER MUST FOLLOW FOR	3
EQUAL SIGN IS NOT IN PROPER POSITION	4
LITERAL PRECEDING TO IS NOT INTEGER	5
IDENTIFIER MUST FOLLOW FOR	6
TO IS MISPLACED OR MISSING	7
POS. INTEGER OR VAR. MUST FOLLOW TO	9
LITERAL FOLLOWING TO IS NOT INTEGER	10
ONLY STEP MAY FOLLOW INTEGER AFTER TO	11
POSITIVE INTEGER MUST FOLLOW STEP	12
LITERAL FOLLOWING STEP IS NOT INTEGER	13
ONLY STEP MAY FOLLOW INTEGER AFTER TO	14
NEXT MUST BE FOLLOWED BY A VARIABLE	15
IMPROPER FOR - NEXT PAIR	16
IMPROPER FOR - NEXT PAIR	17
LINE NUMBER MUST FOLLOW GOSUB STMT	18
MULTIPLE ARGUMENTS IN GOSUB STMT	19
	0
	0
ABEND - SEVERE ERRORS DETECTED.	0
CODE GENERATION SUPPRESSED.	0

APPENDIX D

Model Railroad Control Program

HEX ADDRESS	HEX CONTENTS	MNEMONICS/ DECIMAL CONTENTS	COMMENTS
0000	00	000000000	DATA
0001	20	000000032	DATA
0002	0A	000000010	DATA
0003	80	000000128	DATA
0004	01	000000001	DATA
0005	40	000000064	DATA
0006	02	000000002	DATA
0007	90	000000144	DATA
0008	04	000000004	DATA
0009	00	000000000	DATA
000A	01	000000001	DATA
000B	40	000000064	DATA
000C	01	000000001	CONSTANT
000D	63	000000099	CONSTANT
000E	02	000000002	CONSTANT
000F	00	000000000	CONSTANT
0010	20	000000032	CONSTANT
0011	0A	000000010	CONSTANT
0012	80	000000128	CONSTANT
0013	40	000000064	CONSTANT
0014	90	000000144	CONSTANT
0015	04	000000004	CONSTANT
0016	03	000000003	CONSTANT
0017	FF	000000255	CONSTANT
0018	05	000000005	CONSTANT
0019	A6	000004006	CONSTANT
001A	06	000000006	CONSTANT
001B	07	000000007	CONSTANT
001C	A5	000004005	CONSTANT
001D	08	000000008	CONSTANT
001E	A7	000004007	CONSTANT
001F	09	000000009	CONSTANT
0020	50	000000080	CONSTANT
0021	0B	000000011	CONSTANT
0022	C9	000000201	CONSTANT
0023	0C	000000012	CONSTANT
0024	0D	000000013	CONSTANT
0025	0E	000000014	CONSTANT
0026	0F	000000015	CONSTANT
0027	10	000000016	CONSTANT
0028	11	000000017	CONSTANT
0029	64	000000100	CONSTANT
002A	12	000000018	CONSTANT
002B	13	000000019	CONSTANT
002C	14	000000020	CONSTANT
002D	C8	000000200	CONSTANT
002E	15	000000021	CONSTANT
002F	16	000000022	CONSTANT
0030	17	000000023	CONSTANT
0031	18	000000024	CONSTANT

HEX ADDRESS	HEX CONTENTS	MNEMONICS/ DECIMAL CONTENTS	COMMENTS
0032	19	000000025	CONSTANT
0033	1A	000000026	CONSTANT
0034	1B	000000027	CONSTANT
0035	A4	000004004	CONSTANT
0036	65	000000101	CONSTANT
0037	66	000000102	CONSTANT
0038	CA	000000202	CONSTANT
0039	CB	000000203	CONSTANT
003A	CC	000000204	CONSTANT
003B	CD	000000205	CONSTANT
003C	CE	000000206	CONSTANT
003D	CF	000000207	CONSTANT
003E	D0	000000208	CONSTANT
003F	D1	000000209	CONSTANT
0040	D2	000000210	CONSTANT
0041	D3	000000211	CONSTANT
0042	XX	XXXXXXXXX	D
0043	XX	XXXXXXXXX	C
0044	XX	XXXXXXXXX	S
0045	XX	XXXXXXXXX	P1
0046	XX	XXXXXXXXX	N
0047	XX	XXXXXXXXX	K
0048	XX	XXXXXXXXX	P
0049	XX	XXXXXXXXX	L
004A	XX	XXXXXXXXX	TEMPORARY STORAGE LOCATION
004B	XX	XXXXXXXXX	TEMPORARY STORAGE LOCATION
0100	CE	LDX	RESET INDEX REGISTER TO
0101	00		TOP OF DATA TABLE STARTING
0102	00		AT ADDRESS 0000
0103	96	LDAA	LOAD ACCUMULATOR A DIRECT
0104	17		WITH CONSTANT AT THIS ADDRESS
0105	97	STAA	STORE ACCUMULATOR A DIRECT
0106	42		IN VARIABLE NAME AT THIS ADDRESS
0107	96	LDAA	LOAD DATA STORED IN VARIABLE
0108	42		AT THIS ADDRESS INTO ACCUMULATOR A,
0109	B7	STAA	THEN WRITE IT TO
010A	40		THE OUTPUT DEVICE RESIDING
010B	06		AT THIS HEX ADDRESS
010C	96	LDAA	LOAD ACCUMULATOR A DIRECT
010D	15		WITH CONSTANT AT THIS ADDRESS
010E	97	STAA	STORE ACCUMULATOR A DIRECT
010F	43		IN VARIABLE NAME AT THIS ADDRESS
0110	96	LDAA	LOAD DATA STORED IN VARIABLE
0111	43		AT THIS ADDRESS INTO ACCUMULATOR A,
0112	B7	STAA	THEN WRITE IT TO
0113	40		THE OUTPUT DEVICE RESIDING
0114	05		AT THIS HEX ADDRESS
0115	96	LDAA	LOAD DATA STORED IN VARIABLE
0116	43		AT THIS ADDRESS INTO ACCUMULATOR A,
0117	B7	STAA	THEN WRITE IT TO

HEX ADDRESS	HEX CONTENTS	MNEMONICS/ DECIMAL CONTENTS	COMMENTS
0118	40		THE OUTPUT DEVICE RESIDING
0119	07		AT THIS HEX ADDRESS
011A	96	LDAA	LOAD ACCUMULATOR A DIRECT
011B	20		WITH CONSTANT AT THIS ADDRESS
011C	97	STAA	STORE ACCUMULATOR A DIRECT
011D	44		IN VARIABLE NAME AT THIS ADDRESS
011E	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
011F	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0120	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0121	96	LDAA	LOAD ACCUMULATOR A DIRECT
0122	18		WITH CONSTANT AT THIS ADDRESS
0123	97	STAA	STORE ACCUMULATOR A DIRECT
0124	44		IN VARIABLE NAME AT THIS ADDRESS
0125	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0126	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0127	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0128	96	LDAA	LOAD ACCUMULATOR A DIRECT
0129	0F		WITH CONSTANT AT THIS ADDRESS
012A	97	STAA	STORE ACCUMULATOR A DIRECT
012B	45		IN VARIABLE NAME AT THIS ADDRESS
012C	96	LDAA	LOAD ACCUMULATOR A DIRECT
012D	0C		WITH CONSTANT AT THIS ADDRESS
012E	97	STAA	STORE ACCUMULATOR A DIRECT
012F	46		IN VARIABLE NAME AT THIS ADDRESS
0130	96	LDAA	LOAD FIRST ARGUMENT
0131	46		INTO ACCUMULATOR A
0132	90	SUBA	SUBTRACT SECOND ARGUMENT
0133	23		FROM FIRST ARGUMENT
0134	2F	BLE	IF RESULT OF COMPARISON IS TRUE
0135	03		BRANCH TO NEXT STATEMENT
0136	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0137	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0138	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0139	96	LDAA	LOAD ACCUMULATOR A DIRECT
013A	0C		WITH CONSTANT AT THIS ADDRESS
013B	97	STAA	STORE ACCUMULATOR A DIRECT
013C	47		IN VARIABLE NAME AT THIS ADDRESS
013D	96	LDAA	LOAD FIRST ARGUMENT
013E	47		INTO ACCUMULATOR A
013F	90	SUBA	SUBTRACT SECOND ARGUMENT
0140	29		FROM FIRST ARGUMENT
0141	2F	BLE	IF RESULT OF COMPARISON IS TRUE
0142	03		BRANCH TO NEXT STATEMENT
0143	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0144	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0145	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0146	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0147	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0148	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0149	96	LDAA	LOAD ACCUMULATOR A DIRECT

HEX ADDRESS	HEX CONTENTS	MNEMONICS/ DECIMAL CONTENTS	COMMENTS
014A	47		WITH FIRST OPERAND
014B	9B	ADDA	ADD DIRECT TO ACCUMULATOR A
014C	0C		THE SECOND OPERAND
014D	97	STAA	STORE ACCUMULATOR A DIRECT
014E	4A		IN THIS TEMPORARY LOCATION
014F	96	LDAA	LOAD ACCUMULATOR A DIRECT
0150	4A		WITH TEMP VALUE AT THIS ADDRESS
0151	97	STAA	STORE ACCUMULATOR A DIRECT
0152	47		IN VARIABLE NAME AT THIS ADDRESS
0153	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0154	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0155	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0156	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0157	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0158	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0159	96	LDAA	LOAD ACCUMULATOR A DIRECT
015A	4B		WITH IDENTIFIER VALUE AT THIS ADDRESS
015B	97	STAA	STORE ACCUMULATOR A DIRECT
015C	4A		IN TEMPORARY STORAGE AT THIS ADDRESS
015D	96	LDAA	LOAD ACCUMULATOR A DIRECT
015E	0E		WITH CONSTANT AT THIS ADDRESS
015F	97	STAA	STORE ACCUMULATOR A DIRECT
0160	4B		IN TEMPORARY STORAGE AT THIS ADDRESS
0161	96	LDAA	LOAD FIRST ARGUMENT
0162	4A		INTO ACCUMULATOR A
0163	90	SUBA	SUBTRACT SECOND ARGUMENT
0164	4B		FROM FIRST ARGUMENT
0165	27	BEQ	IF RESULT OF COMPARISON IS TRUE
0166	03		BRANCH TO NEXT STATEMENT
0167	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0168	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0169	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
016A	96	LDAA	LOAD ACCUMULATOR A DIRECT
016B	0C		WITH CONSTANT AT THIS ADDRESS
016C	97	STAA	STORE ACCUMULATOR A DIRECT
016D	45		IN VARIABLE NAME AT THIS ADDRESS
016E	96	LDAA	LOAD ACCUMULATOR A DIRECT
016F	4B		WITH IDENTIFIER VALUE AT THIS ADDRESS
0170	97	STAA	STORE ACCUMULATOR A DIRECT
0171	4A		IN TEMPORARY STORAGE AT THIS ADDRESS
0172	96	LDAA	LOAD ACCUMULATOR A DIRECT
0173	0C		WITH CONSTANT AT THIS ADDRESS
0174	97	STAA	STORE ACCUMULATOR A DIRECT
0175	4B		IN TEMPORARY STORAGE AT THIS ADDRESS
0176	96	LDAA	LOAD FIRST ARGUMENT
0177	4A		INTO ACCUMULATOR A
0178	90	SUBA	SUBTRACT SECOND ARGUMENT
0179	4B		FROM FIRST ARGUMENT
017A	27	BEQ	IF RESULT OF COMPARISON IS TRUE
017B	03		BRANCH TO NEXT STATEMENT

HEX ADDRESS	HEX CONTENTS	MNEMONICS/ DECIMAL CONTENTS	COMMENTS
017C	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
017D	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
017E	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
017F	96	LDAA	LOAD ACCUMULATOR A DIRECT
0180	0F		WITH CONSTANT AT THIS ADDRESS
0181	97	STAA	STORE ACCUMULATOR A DIRECT
0182	45		IN VARIABLE NAME AT THIS ADDRESS
0183	96	LDAA	LOAD ACCUMULATOR A DIRECT
0184	46		WITH FIRST OPERAND
0185	9B	ADDA	ADD DIRECT TO ACCUMULATOR A
0186	0C		THE SECOND OPERAND
0187	97	STAA	STORE ACCUMULATOR A DIRECT
0188	4A		IN THIS TEMPORARY LOCATION
0189	96	LDAA	LOAD ACCUMULATOR A DIRECT
018A	4A		WITH TEMP VALUE AT THIS ADDRESS
018B	97	STAA	STORE ACCUMULATOR A DIRECT
018C	46		IN VARIABLE NAME AT THIS ADDRESS
018D	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
018E	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
018F	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0190	CE	LDX	RESET INDEX REGISTER TO
0191	00		TOP OF DATA TABLE STARTING
0192	00		AT ADDRESS 0000
0193	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0194	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0195	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
0196	3E	WAI	HALT EXECUTION (WAIT FOR INTERRUPT)
0197	B6	LDAA	LOAD DATA FROM INPUT
0198	40		DEVICE AT THIS ADDRESS
0199	04		INTO ACCUMULATOR A. THEN
019A	97	STAA	STORE DATA IN VARIABLE
019B	4B		NAME AT THIS ADDRESS
019C	96	LDAA	LOAD ACCUMULATOR A DIRECT
019D	4B		WITH IDENTIFIER VALUE AT THIS ADDRESS
019E	97	STAA	STORE ACCUMULATOR A DIRECT
019F	4A		IN TEMPORARY STORAGE AT THIS ADDRESS
01A0	96	LDAA	LOAD ACCUMULATOR A DIRECT
01A1	45		WITH IDENTIFIER VALUE AT THIS ADDRESS
01A2	97	STAA	STORE ACCUMULATOR A DIRECT
01A3	4B		IN TEMPORARY STORAGE AT THIS ADDRESS
01A4	96	LDAA	LOAD FIRST ARGUMENT
01A5	4A		INTO ACCUMULATOR A
01A6	90	SUBA	SUBTRACT SECOND ARGUMENT
01A7	4B		FROM FIRST ARGUMENT
01A8	26	BNE	IF RESULT OF COMPARISON IS TRUE
01A9	03		BRANCH TO NEXT STATEMENT
01AA	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01AB	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01AC	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01AD	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER

HEX ADDRESS	HEX CONTENTS	MNEMONICS/ DECIMAL CONTENTS	COMMENTS
01AF	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01AF	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01B0	39	RTS	RETURN FROM SUBROUTINE
01B1	A6	LDAA, X	LOAD DATA INTO ACCUMULATOR A USING
01B2	00		INDEXED ADDRESSING WITH ZERO OFFSET
01B3	97	STAA	STORE DATA DIRECT INTO VARIABLE NAME
01B4	44		AT THIS ZERO PAGE ADDRESS
01B5	08	INX	INCREMENT INDEX REGISTER
01B6	96	LDAA	LOAD DATA STORED IN VARIABLE
01B7	44		AT THIS ADDRESS INTO ACCUMULATOR A,
01B8	B7	STAA	THEN WRITE IT TO
01B9	40		THE OUTPUT DEVICE RESIDING
01BA	06		AT THIS HEX ADDRESS
01BB	86	LDAA	LOAD ACCUMULATOR A DIRECT
01BC	0C		WITH CONSTANT AT THIS ADDRESS
01BD	97	STAA	STORE ACCUMULATOR A DIRECT
01BE	47		IN VARIABLE NAME AT THIS ADDRESS
01BF	96	LDAA	LOAD FIRST ARGUMENT
01C0	47		INTO ACCUMULATOR A
01C1	90	SUBA	SUBTRACT SECOND ARGUMENT
01C2	10		FROM FIRST ARGUMENT
01C3	2F	BLE	IF RESULT OF COMPARISON IS TRUE
01C4	03		BRANCH TO NEXT STATEMENT
01C5	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01C6	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01C7	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01C8	96	LDAA	LOAD ACCUMULATOR A DIRECT
01C9	0C		WITH CONSTANT AT THIS ADDRESS
01CA	97	STAA	STORE ACCUMULATOR A DIRECT
01CB	49		IN VARIABLE NAME AT THIS ADDRESS
01CC	96	LDAA	LOAD FIRST ARGUMENT
01CD	49		INTO ACCUMULATOR A
01CE	90	SUBA	SUBTRACT SECOND ARGUMENT
01CF	10		FROM FIRST ARGUMENT
01D0	2F	BLE	IF RESULT OF COMPARISON IS TRUE
01D1	03		BRANCH TO NEXT STATEMENT
01D2	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01D3	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01D4	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01D5	96	LDAA	LOAD ACCUMULATOR A DIRECT
01D6	0F		WITH CONSTANT AT THIS ADDRESS
01D7	97	STAA	STORE ACCUMULATOR A DIRECT
01D8	44		IN VARIABLE NAME AT THIS ADDRESS
01D9	96	LDAA	LOAD ACCUMULATOR A DIRECT
01DA	0F		WITH CONSTANT AT THIS ADDRESS
01DB	97	STAA	STORE ACCUMULATOR A DIRECT
01DC	44		IN VARIABLE NAME AT THIS ADDRESS
01DD	96	LDAA	LOAD ACCUMULATOR A DIRECT
01DE	49		WITH FIRST OPERAND
01DF	9B	ADDA	ADD DIRECT TO ACCUMULATOR A

HEX ADDRESS	HEX CONTENTS	MNEMONICS/ DECIMAL CONTENTS	COMMENTS
01E0	0C		THE SECOND OPERAND
01E1	97	STAA	STORE ACCUMULATOR A DIRECT
01E2	4A		IN THIS TEMPORARY LOCATION
01E3	96	LDAA	LOAD ACCUMULATOR A DIRECT
01E4	4A		WITH TEMP VALUE AT THIS ADDRESS
01E5	97	STAA	STORE ACCUMULATOR A DIRECT
01E6	49		IN VARIABLE NAME AT THIS ADDRESS
01E7	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01E8	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01E9	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01EA	96	LDAA	LOAD ACCUMULATOR A DIRECT
01EB	47		WITH FIRST OPERAND
01EC	9B	ADDA	ADD DIRECT TO ACCUMULATOR A
01ED	0C		THE SECOND OPERAND
01EE	97	STAA	STORE ACCUMULATOR A DIRECT
01EF	4A		IN THIS TEMPORARY LOCATION
01F0	96	LDAA	LOAD ACCUMULATOR A DIRECT
01F1	4A		WITH TEMP VALUE AT THIS ADDRESS
01F2	97	STAA	STORE ACCUMULATOR A DIRECT
01F3	47		IN VARIABLE NAME AT THIS ADDRESS
01F4	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01F5	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01F6	XX	XXXXXXXXX	BRANCH INSTRUCTION - TO BE FILLED IN LATER
01F7	96	LDAA	LOAD DATA STORED IN VARIABLE
01F8	44		AT THIS ADDRESS INTO ACCUMULATOR A.
01F9	B7	STAA	THEN WRITE IT TO
01FA	40		THE OUTPUT DEVICE RESIDING
01FB	0A		AT THIS HEX ADDRESS
01FC	39	RTS	RETURN FROM SUBROUTINE
01FD	3E	WAI	HALT EXECUTION (WAIT FOR INTERRUPT)
011E	BD	JSR	JUMP TO SUBROUTINE
011F	01		BEGINNING AT THIS
0120	B6		MEMORY LOCATION
0125	BD	JSR	JUMP TO SUBROUTINE
0126	01		BEGINNING AT THIS
0127	B6		MEMORY LOCATION
0136	7E	JMP	JUMP TO
0137	01		THIS
0138	90		HEX ADDRESS
0143	7E	JMP	JUMP TO
0144	01		THIS
0145	56		HEX ADDRESS
0146	BD	JSR	JUMP TO SUBROUTINE
0147	01		BEGINNING AT THIS
0148	97		MEMORY LOCATION
0153	7E	JMP	JUMP TO
0154	01		THIS
0155	30		HEX ADDRESS
0156	BD	JSR	JUMP TO SUBROUTINE
0157	01		BEGINNING AT THIS

HEX ADDRESS	HEX CONTENTS	MNEMONICS/ DECIMAL CONTENTS	COMMENTS
0166	81		MEMORY LOCATION
0167	7E	JMP	JUMP TO
0168	01		THIS
0169	6E		HEX ADDRESS
016C	7E	JMP	JUMP TO
016D	01		THIS
016E	83		HEX ADDRESS
0170	7E	JMP	JUMP TO
0171	01		THIS
0172	30		HEX ADDRESS
0176	7E	JMP	JUMP TO
0177	01		THIS
0178	2C		HEX ADDRESS
017B	7E	JMP	JUMP TO
017C	01		THIS
017D	B0		HEX ADDRESS
0180	7E	JMP	JUMP TO
0181	01		THIS
0182	97		HEX ADDRESS
0185	7E	JMP	JUMP TO
0186	01		THIS
0187	E7		HEX ADDRESS
0189	7E	JMP	JUMP TO
018A	01		THIS
018B	FA		HEX ADDRESS
018E	7E	JMP	JUMP TO
018F	01		THIS
0190	CC		HEX ADDRESS
0193	7E	JMP	JUMP TO
0194	01		THIS
0195	BE		HEX ADDRESS

PROGRAM STARTING ADDRESS 0100

TOTAL MEMORY USED : 510 BYTES

SOURCE LISTING

```
1     REM THIS PROGRAM DRIVES HO R.R. SWITCHES                    1
2     DATA 0, 32, 10, 128, 1, 64, 2, 144, 4, 0, 1, 64             2
3     REM INITIALIZE PIA                                          3
4     LET D = 255                                                 4
5     WRTOUT D TO 4006                                            5
6     LET C = 4                                                   6
7     WRTOUT C TO 4005                                            7
8     WRTOUT C TO 4007                                            8
9     REM SET ALL SWITCHES STRAIGHT THROUGH                       9
10    LET S = 80                                                  10
11    GOSUB 201                                                   11
12    LET S = 5                                                   12
13    GOSUB 201                                                   13
14    REM BEGIN MAIN PROGRAM                                      14
15    LET P1 = 2                                                  15
16    FOR N = 1 TO 12                                             16
17    FOR K = 1 TO 100                                            17
18    GOSUB 100                                                   18
19    NEXT K                                                      19
20    GOSUB 200                                                   20
21    IF P = 2 THEN LET P1 = 1                                    21
22    IF P = 1 THEN LET P1 = 2                                    22
23    NEXT N                                                      23
24    RESTORE                                                     24
25    GOTO 16                                                     25
26    STOP                                                        26
27    REM SUBROUTINES                                             27
100   READ P FROM 4004                                            28
101   IF P NEQ P1 GOTO 100                                        29
102   RETURN                                                      30
200   READ S                                                      31
201   WRTOUT S TO 4006                                            32
202   REM DELAY FOR SWITCH RESPONSE                               33
203   FOR K = 1 TO 32                                             34
204   FOR L = 1 TO 32                                             35
205   LET S = 0                                                   36
206   LET S = 0                                                   37
207   NEXT L                                                      38
208   NEXT K                                                      39
209   WRTOUT S TO 4006                                            40
210   RETURN                                                      41
211   END                                                         42
```

ERROR TABLE

ERROR	SOURCE LINE
NO DIAGNOSTICS GENERATED	0
	0
MAXIMUM TEMPORARY STORAGE USED -	2

KEYWORD TABLE

LET	1
GOTO	2
IF	3
TO	4
THEN	5
FOR	6
NEXT	7
GOSUB	8
RETURN	9
STEP	10
REM	11
DATA	12
READ	13
RESTORE	14
FROM	15
STOP	16
END	17
WRTOUT	18
RDIN	19

TERMINAL TABLE

-	1
+	2
/	3
*	4
↑	5
=	6
(	7
)	8
>	9
<	10
,	11
.	12
ABS	13
ATN	14
COS	15
DEF	16
EXP	17
INT	18
LOG	19
SIN	20
SQR	21
TAN	22
$	23
LEQ	24
GEQ	25
NEQ	26
CEQ	27

IDENTIFIER TABLE

D	1
C	2
S	3
P1	4
N	5
K	6
P	7
L	*

LITERAL TABLE

1	1
99	2
2	3
0	4
32	5
10	6
128	7
64	8
144	9
4	10
3	11
255	12
5	13
4006	14
6	15
7	16
4005	17
8	18
4007	19
9	20
90	21
11	22
201	23
12	24
13	25
14	26
15	27
16	28
17	29
100	30
18	31
19	32
20	33
200	34
21	35
22	36
23	37
24	38
25	39
26	40
27	41
4004	42
101	43
102	44
20	45
203	46
204	47
205	48
206	49
207	50
208	51
209	52
210	53
211	54

UNIFORM SYMBOL TABLE

SYMBOL	POINTER
LIT	1
TRM	23
LIT	3
KEY	12
LIT	4
TRM	11
LIT	5
TRM	11
LIT	6
TRM	11
LIT	7
TRM	11
LIT	1
TRM	11
LIT	8
TRM	11
LIT	3
TRM	11
LIT	9
TRM	11
LIT	10
TRM	11
LIT	4
TRM	11
LIT	1
TRM	11
LIT	8
TRM	23
LIT	11
TRM	23
LIT	10
KEY	1
IDN	1
TRM	6
LIT	12
TRM	23
LIT	13
KEY	18
IDN	1
KEY	4
LIT	14
TRM	23
LIT	15
KEY	1
IDN	2
TRM	6
LIT	10
TRM	23
LIT	16
KEY	18
IDN	2
KEY	4
LIT	17
TRM	23
LIT	18
KEY	18

IDN	2
KEY	4
LIT	19
TRM	23
LIT	20
TRM	23
LIT	6
KEY	1
IDN	3
TRM	6
LIT	21
TRM	23
LIT	22
KEY	8
LIT	23
TRM	23
LIT	24
KEY	1
IDN	3
TRM	6
LIT	13
TRM	23
LIT	25
KEY	8
LIT	23
TRM	23
LIT	26
TRM	23
LIT	27
KEY	1
IDN	4
TRM	6
LIT	3
TRM	23
LIT	28
KEY	6
IDN	5
TRM	6
LIT	1
KEY	4
LIT	24
TRM	23
LIT	29
KEY	6
IDN	6
TRM	6
LIT	1
KEY	4
LIT	30
TRM	23
LIT	31
KEY	8
LIT	30
TRM	23
LIT	32
KEY	7
IDN	6
TRM	23
LIT	33
KEY	8

LIT	34
TRM	23
LIT	35
KEY	3
IDN	7
TRM	6
LIT	3
KEY	5
KEY	1
IDN	4
TRM	6
LIT	1
TRM	23
LIT	36
KEY	3
IDN	7
TRM	6
LIT	1
KEY	5
KEY	1
IDN	4
TRM	6
LIT	3
TRM	23
LIT	37
KEY	7
IDN	5
TRM	23
LIT	38
KEY	14
TRM	23
LIT	39
KEY	2
LIT	28
TRM	23
LIT	40
KEY	16
TRM	23
LIT	41
TRM	23
LIT	30
KEY	19
IDN	7
KEY	15
LIT	42
TRM	23
LIT	43
KEY	3
IDN	7
TRM	26
IDN	4
KEY	2
LIT	30
TRM	23
LIT	44
KEY	9
TRM	23
LIT	34
KEY	13
IDN	3

TRM	23
LIT	23
KEY	18
IDN	3
KEY	4
LIT	14
TRM	23
LIT	45
TRM	23
LIT	46
KEY	6
IDN	6
TRM	6
LIT	1
KEY	4
LIT	5
TRM	23
LIT	47
KEY	6
IDN	8
TRM	6
LIT	1
KEY	4
LIT	5
TRM	23
LIT	48
KEY	1
IDN	3
TRM	6
LIT	4
TRM	23
LIT	49
KEY	1
IDN	3
TRM	6
LIT	4
TRM	23
LIT	50
KEY	7
IDN	8
TRM	23
LIT	51
KEY	7
IDN	6
TRM	23
LIT	52
KEY	18
IDN	3
KEY	4
LIT	14
TRM	23
LIT	53
KEY	9
TRM	23
LIT	54
KEY	17
TRM	23

DATA TABLE

LIT TAL POINTER
4
5
6
7
1
8
3
9
10
4
1
8

FILE: MATRIX DATA A YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

```
0010TRM0006IDN0001LIT0012
0013KEY0018IDN0001LIT0014
0015TRM0004IDN0002LIT0010
0016KEY0019IDN0002LIT0017
0018KEY0018IDN0002LIT0019
0006TRM0004IDN0003LIT0021
0022KEY0008LIT0023    0000
0024TRM0006IDN0003LIT0013
0025KEY0008LIT0023    0000
0027TRM0006IDN0004LIT0003
0028TRM0006IDN0005LIT0001
0028TRM0024IDN0005LIT0024
0028KEY0002LIT0038    0000
0029TRM0006IDN0006LIT0001
0029TRM0024IDN0006LIT0030
0029KEY0002LIT0033    0000
0031KEY0008LIT0030    0000
0032TRM0002IDN0006LIT0001
0032TRM0006IDN0006TMP0001
0032KEY0002LIT0029LIT0002
0033KEY0008LIT0034    0000
0035TRM0006TMP0001IDN0007
0035TRM0006TMP0002LIT0003
0035TRM0027TMP0001TMP0002
0035KEY0002LIT0036    0000
0035TRM0006IDN0004LIT0001
0036TRM0006TMP0001IDN0007
0036TRM0006TMP0002LIT0001
0036TRM0027TMP0001TMP0002
0036KEY0002LIT0037    0000
0036TRM0006IDN0004LIT0003
0037TRM0002IDN0005LIT0001
0037TRM0005IDN0005TMP0001
0037KEY0002LIT0038LIT0002
0038KEY0014    0000    0000
0039KEY0002LIT0028    0000
0040KEY0015    0000    0000
0030KEY0019IDN0007LIT0042
0041TRM0006TMP0001IDN0007
0043TRM0006TMP0002IDN0004
0043TRM0026TMP0001TMP0002
0043KEY0002LIT0044    0000
0043KEY0002LIT0030    0000
0044KEY0009    0000    0000
0034KEY0013IDN0003    0000
0023KEY0018IDN0003LIT0014
0046TRM0006IDN0006LIT0001
0046TRM0024IDN0006LIT0005
0046KEY0002LIT0052    0000
0047TRM0006IDN0004LIT0001
0047TRM0024IDN0004LIT0005
0047KEY0002LIT0051    0000
0048TRM0006IDN0003LIT0004
0049TRM0006IDN0003LIT0004
0050TRM0002IDN0004LIT0001
```

```
FILE: MATRIX    DATA      A                YOUNGSTOWN STATE UNIVERSITY COMPUTER CENTER

0050TRM0006IDN0008TMP0001
0050KEY0002LIT0047LIT0002
0051TRM0002IDN0006LIT0001
0051TRM0006IDN0006TMP0001
0051KEY0002LIT0046LIT0002
0052KEY0018IDN0003LIT0014
0053KEY0009     0000     0000
0054KEY0017     0000     0000
```

APPENDIX E

Glossary of Compiler Terms

GLOSSARY OF COMPILER TERMS [4]

ACTION ROUTINE. Interprets the meaning of basic syntax constructions and generates matrix entries.

CODE GENERATION. Third and final phase of the compiler. Produces appropriate microprocessor code.

COMPILER. Accepts a program written in a higher level language as input and produces its machine equivalent as output.

IDENTIFIER TABLE. Created by lexical analysis. Contains all variables in the program.

KEYWORD TABLE. Permanent table. Contains all keywords for modified BASIC language.

LEXICAL ANALYSIS. Recognition of basic elements and creation of uniform symbols.

LITERAL TABLE. Created by lexical analysis. Contains all constants in the source program.

MATRIX. Intermediate form of the program which is created by action routines, and is then used for code generation.

PARSE TABLE. A list of the tokens as they appear in the source program. Created by lexical analysis.

SOURCE PROGRAM. The higher level language program used as input to the compiler.

SYNTAX ANALYSIS. Recognition of basic constructs and associated meanings according to the rules of syntax for the source language.

TERMINAL TABLE. Permanent table. Lists all the operators and special symbols for the source language.

TOKEN. Basic elements of the source program. They are identifiers, literals, terminal symbols, and keywords which are delineated by blanks, operators, and special symbols.

UNIFORM SYMBOL TABLE. Abbreviated form of the parse table with each entry containing the identification of the table to which it belongs and its index in that table.

REFERENCES

[1]. Individual Learning Program-Microprocessors, Heathkit Continuing Education Series, Heath Company, Benton Harbor, Michigan, 1977.

[2]. Heathkit Manual for the Microprocessor Trainer Model ET-3400, Heath Company, Benton Harbor, Michigan, 1977.

[3]. BASIC-PLUS Language Manual, Digital Equipment Corporation, Maynard, Mass., May 1971; 4th revision May 1975.

[4]. Donovan, John J., Systems Programming, McGraw-Hill Book Company, New York, 1972.

[5]. IBM Virtual Machine Facility/370: CMS User's Guide, International Business Machines Corporation, Poughkeepsie, New York, 1976.