Computer Simulation:
A Simulation Language and Example

By

Ralph Edwin Love, Jr.

B.S. (Stanford University) 1957

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA

Approved:

[Signatures]

Committee in Charge

 Deposited in the University Library... JUL 20 1962
 Date Librarian
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <strong>Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>2. <strong>BC NELIAC</strong></td>
<td>4</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td></td>
</tr>
<tr>
<td>2.2 Metalanguage</td>
<td>5</td>
</tr>
<tr>
<td>2.3 Flowchart</td>
<td>6</td>
</tr>
<tr>
<td>2.4 Declaration Lists</td>
<td>7</td>
</tr>
<tr>
<td>2.5 Variables</td>
<td>10</td>
</tr>
<tr>
<td>2.6 Expressions</td>
<td>10</td>
</tr>
<tr>
<td>2.7 Program Logic</td>
<td>12</td>
</tr>
<tr>
<td>2.7.1 Assignment Statements</td>
<td>13</td>
</tr>
<tr>
<td>2.7.2 GO TO Statements and SWITCH Statements</td>
<td>13</td>
</tr>
<tr>
<td>2.7.3 FOR Statements</td>
<td>14</td>
</tr>
<tr>
<td>2.7.4 DO Statements</td>
<td>14</td>
</tr>
<tr>
<td>2.7.5 Conditional Statements</td>
<td>16</td>
</tr>
<tr>
<td>3. <strong>Intercom 500</strong></td>
<td>17</td>
</tr>
<tr>
<td>3.1 Intercom 500 Computer</td>
<td>17</td>
</tr>
<tr>
<td>3.1.1 Computer Organization</td>
<td>17</td>
</tr>
<tr>
<td>3.1.2 Command Structure</td>
<td>20</td>
</tr>
<tr>
<td>3.1.3 Operation Codes</td>
<td>20</td>
</tr>
<tr>
<td>3.2 Algorithm for Simulation</td>
<td>22</td>
</tr>
<tr>
<td>3.3 Conclusions</td>
<td>23</td>
</tr>
<tr>
<td>4. <strong>Symbolic Intercom 500</strong></td>
<td>36</td>
</tr>
<tr>
<td>4.1 Source Language</td>
<td>36</td>
</tr>
<tr>
<td>4.2 Algorithm</td>
<td>38</td>
</tr>
</tbody>
</table>
## TABLE OF CONTENTS (cont.)

Appendix

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Transliteration Rules</td>
<td>40</td>
</tr>
<tr>
<td>B. Operation Code Limitations</td>
<td>42</td>
</tr>
<tr>
<td>C. Intercom Card Format</td>
<td>44</td>
</tr>
<tr>
<td>D. Use of Machine Language Subroutines in Symbolic Intercom</td>
<td>45</td>
</tr>
<tr>
<td>E. Symbolic Intercom 500 Assembler</td>
<td>47</td>
</tr>
<tr>
<td>F. BC NELIAC Simulation of Intercom 500</td>
<td>48</td>
</tr>
<tr>
<td>G. A Syntactical Flowchart for BC NELIAC</td>
<td>49</td>
</tr>
</tbody>
</table>

Bibliography                                                           | 50   |
Chapter 1

INTRODUCTION

The term "simulation" can be defined as the replacement of a given system by a substitute system, or "simulator", which responds to the external environment in a similar way as the original system. With the development of large scale data processors "simulation" of systems has become a field of important interest and study. Simulation studies have been made on such subjects as the nation's economy, mental activities of the brain and new digital computer systems. As digital computers become larger and faster, simulations will become more accurate and complex in their representation of the original system. In all simulation problems run on a computer it is necessary to write a program, which consists of a set of instructions to the computer, to direct the machine's operation and perform the desired simulation.

A program may be written in the following forms: machine-language coding, assembly language, or problem oriented languages. In machine language coding each individual instruction is written in the numerical language of the specific computer for which the program was intended. The assembly language allows the programmer to refer to computer functions or to memory storage addresses symbolically, with letters instead of numbers. Problem oriented languages allow the computer user to express a program in terms of the problem, instead of the computer.
To write programs in machine or assembly languages requires the programmer know most of the machine or symbolic instructions and their various ways of being modified. Since the program has to be written on an instruction-by-instruction basis, the process of developing a program can be complex, tedious, and slow.

Problem oriented languages allow the computer user freedom to think in terms of the problem and less in terms of the details of the computer. If the language used is machine independent, then the programs written in the problem oriented language will not become outdated as new computers are developed and marketed.

The purpose of this paper is to describe a machine independent, problem oriented language, BC NELIAC, which developed into being a useful language for writing programs for simulation problems. The most significant feature of BC NELIAC is that programs coded in the source language are self-documenting. Four other important features which make BC NELIAC useful for simulation problems are partword operations, chain expressions, ALGOL type source language, and fast compiling speed. Partword operations and chain expressions are explained in the section on BC NELIAC. Samples of the BC NELIAC source language are shown throughout this paper and in the appendix.

As a simulation example, the programming language, Intercom 500, used on the Bendix G-15 digital computer will be described. This simulation problem was chosen for two purposes: first, to consider the economy of operating a small scale Intercom Computer versus a large scale data processor simulating
Intercom 500, and second, to provide Intercom on a machine with a larger memory and faster operating speed. As an introduction to the paper a description of BC NELIAC will be given after which an explanation of Intercom 500 will be developed. Finally, the algorithm used in the simulation will be shown and a modification to the Intercom 500 language will be presented.
Chapter 2

BC NELIAC

2.1 Introduction

BC NELIAC is a problem oriented language which is used on the IBM 704 large-scale general purpose digital computer. It was developed by students at the University of California and is a modified version of the original Neliac created at the Naval Electronics Laboratory in San Diego, California. The most significant feature of Neliac is that the translator is written in its own language. This feature has allowed modifications to be made to Neliac quickly and easily. One of the main distinctions of BC NELIAC is the inclusion of some ALGOL-60 delimiters to the source language.¹

The delimiters not only make the program more readable but allow the programmer greater ease in visualizing and writing the simulation program. Another important feature is chain expressions which has been added to BC NELIAC to simplify operations involving character manipulations and make the language less machine oriented.² This is an account of the more important features of BC NELIAC. The aim is to explain rather than to define the language.

¹ The ALGOL 60 delimiters were added to the Neliac Load Source flow chart in the summer, 1961, by Ralph Love.

² Chain expressions were added to the Neliac system in January 1962, by Niklaus Wirth.
2.2 Metalanguage

The syntax of BC NELIAC is described using the ALGOL metalanguage. It will be helpful to use this metalanguage in the following explanation of BC NELIAC. The basic symbols of this language are:

\[::=\text{Metalinguistic connective meaning "is defined to be"}\]
\[
|\text{Metalinguistic connective meaning "or"}\]
\[
< > \text{Delimiting brackets which enclose metalinguistic variables.}\]

Metalinguistic variables are a sequence of characters enclosed in the delimiting brackets \(< >\). The symbols used for distinguishing the metalinguistic variables have been chosen to be words describing approximately the nature of the corresponding variable. This is done only for understanding and has no technical significance. In a formula a mark, which is not a variable, connective, or a delimiter denotes itself. Juxtaposition of marks and/or variables in a formula signifies juxtaposition of the marks and/or variables in the language being defined. Metalinguistic formulae are composed of metalinguistic connectives, variables enclosed within delimiting brackets, and an indication of juxtaposition.

Metalinguistic Formula example.

\[
<\text{identifier}>::=<\text{letter}|\text{identifier}\>|\text{letter}>
\]
\[
<\text{identifier}|\text{digit}>\]
\[
<\text{letter}>::=\text{A}|\text{B}|\text{C}|\ldots|\text{Y}|\text{Z}
\]
\[
<\text{digit}>::=0|1|2|\ldots|8|9
\]
The formula for identifier is recursive since identifier appears on both sides of the "defining connective." The metalinguistic variable letter indicates identifier can have the value A, or B, or C, etc. The marks identifier digit mean given some value of identifier another can be formed by juxtapositioning a value of the variable digit.

If the values of digit are the arabic numerals then the following are illustrations of legitimate values of identifier:

A
AB
ALB
Y55A
XYZ799

The BC NELIAC reference language will be used in the program examples of this paper. In some cases the symbols used in the reference language are not available in the character set used with the IBM 704 digital computer. Rules for transliteration from the reference language to the hardware representation are included in the appendix.

2.3 Flowchart

The logical segment of the BC NELIAC program is the flowchart. It consists of two parts; the first part is a declaration or dimension list and the second part is the program logic. In the declaration portion variables are declared and in some cases set equal to initial values. The program logic portion is the actual program which specifies the operations to be
performed on the variables defined in the declaration list. The program logic consists of a sequence of statements, which are separated by punctuation marks (usually commas.) By labelling single statements, with an identifier and a colon, they may be referred to from other points of the program.

Normally statements will be executed consecutively. This rule may be broken by introducing **GO TO** statements which explicitly specify the next statement to be executed, or **DO** statements which cause a subroutine to be executed and then control returned to the statement after the **DO** statement. The processing sequence of the program may be shortened by conditional statements, which may cause certain statements to be skipped.

If the program logic portion of a flowchart is called a compound tail then a flowchart has the form:

\[
\langle \text{flowchart} \rangle ::= \langle \text{declaration list} \rangle ; \langle \text{compound tail} \rangle ..
\]

A BC NELIAC program consists of a sequence of flowcharts. The flowcharts are not independent logical segments. Variables declared or labels occurring in any flowchart may be referred to from within any arbitrary flowchart; however, normally variables should be declared before they are called.

Flowchart example

\[
A, B, C;
\]

\[
\text{SUM: } A + B + C \to A ..
\]

2.4 Declaration Lists

All variables used in the program except labels and indices must be declared. Declaration lists serve to define certain
properties of the variables in the program. Declaration of a variable may consist of a declaration identifier, alternate name, structure declaration, and value list. Each declaration is separated by a comma in the declaration list.

A declaration identifier is the name by which the declared variable will be referred. If more than one name is given to the identical variable, alternate names may be listed with a colon in between.

A variable is normally a computer word (36 bits for the IBM 704 Computer); however, the structure declaration contains information about the sub-structure of the variable, which may consist of several part words or a chain of characters. Names referring to partwords are included within a left brace and right brace in the declaration. Each partword name is followed by a definition of the part or subfield of the computer word it represents and is enclosed in parentheses. The partword limits specify the right most (lowest) and the left most (highest) bit belonging to the named partword.

A variable may consist of a chain of characters, symbols, or groups of bits which in the program will be treated as separate entities in the program logic. The structure declaration for a chain variable consists of the number of bits forming a character or symbol preceded by an asterisk and enclosed in parenthesis.

The value list may pre-assign a numerical value to a variable and/or define the dimension of a variable in the case
of an array.

The value list consists of two parts, both of which may be empty. The first part defines the dimensions of the variable in the case of an array (if it is empty, the dimension is assumed to be 1) The second part is the number list in the case of an array, which degenerates to a number in the case of a single variable. If the number list is empty, the variable is pre-assigned the value 0.

Also, a variable may be assigned a predetermined location in the IBM 704 computer (absolute addressing), by following the variable with * OCT and an octal integer.

The declaration of a variable can have the following form:

```
declaration ::= ⟨declaration identifier⟩
⟨structure declaration⟩⟨alternate names⟩
⟨value list⟩
```

Declaration Examples

Simple Variables

A, B, C

Alternate Names

A: A1: A2;
A: B: C4,

Partword

A: \{B(7 \rightarrow 10), C(9 \rightarrow 12)\},

Chain Variable

A (*6), B(*9),

Value List Assigned to Variable

A \leftarrow 5, B(3) \leftarrow 2, 1, 5,
Array
A(10), B(5),

Declaration List
A, B, C,

BLOCK OF WORDS (100),

WORD: ALTERNATE NAME 1: ALTERNATE NAME 2,

INSTRUCTION: 
{ PREFIX (33 \rightarrow 35), DECREMENT (18 \rightarrow 32),
  TAG (15 \rightarrow 17), ADDRESS (0 \rightarrow 14),
  RIGHT WORD (0 \rightarrow 17) }

2.5 Variables

Variables are combined with numbers, punctuation, and operational symbols to form expressions and statements. Variables can be declared as fixed point or floating point quantities. A subscripted variable designates values which are components of linear or single dimensional arrays. The array components of linear or single dimensional arrays. The array component referred to by a subscripted variable is specified by the actual numerical value of the subscript expression and will be an integer.

The letters I through N are reserved for variables of a particular type known as indices, and they must not be declared.

2.6 Expressions

Expressions are the major constituents of statements. There are five important types of expressions used in BC NELIAC. They are: arithmetic expressions, Boolean expressions, designational expressions, chain expressions, and logical expressions.
Arithmetic expressions are used to compute a numerical value by executing the indicated arithmetic operations on the actual numerical values of the variables of the expressions. The arithmetic expression is followed by a left to right arrow to denote replacement and a variable which is set equal to that which preceded the arrow.

Boolean expressions consist of a comparison of an arithmetic expression and a variable. Boolean expressions produce an output of true or false, depending on whether the condition stated is satisfied, or not.

Designational expressions may be either a label or a switch designator which consists of a label and subscript. They are normally used in GO TO statements.

Chain expressions are intended to simplify operations involving character manipulations. A variable will consist of a chain of characters when a chain declaration is applied to it. Two operations may be performed on chain variables—"catenate" and "obtain first character". The "catenating operation" will left shift a chain variable one character and add at the right another character. Its form is:

\[ \text{Variable} \ 1 \ + \ + \ \text{Variable} \ 2 \]

The "obtain first character operation" will obtain the left most character of a chain variable, and has the form:

\[ \ast \text{Variable} \ \text{binary operator} \]

The logical AND or OR functions of two variables is performed using the logical expression.
2.7 Program Logic

The program logic or compound tail portion of a flowchart consists of statements which are the unit of instructions, or sentences, of this algebraic language. As in written English their order of appearance is important. Statements may be chained together with commas in between thus forming unconditional statements, or they may be prefixed by conditions, thus forming conditional statements.

A compound statement may be formed by grouping a set of statements together with BEGIN preceding the first statement and END following the last statement. Any statement within a compound statement may itself be a compound statement.

A portion of the syntax for the program logic section is:

\[
\text{<compound tail> ::= <statement> | <statement> <compound tail>}
\]

\[
\text{<statement> ::= <label>: <statement> | unconditional statement}> | <conditional statement>
\]

\[
\text{<compound statement> ::= BEGIN <compound tail> END}
\]

There are six important types of statements which will be discussed. They are assignment statements, GO TO statements, SWITCH statements, FOR statements, DO statements and conditional statements. The first five of these are considered unconditional statements.
2.7.1 Assignment Statements

The assignment statement specifies an expression to be evaluated and a variable which is to have the resulting value assigned to it. If the variable to the right of an arrow is designating a partial word, then the part(s) of the word not designated remain unaffected by the assignment statement.

An assignment statement is executed in the following steps:

1) the expression to the left of the arrow is evaluated
2) the subscript expression of the variable to the right of the left most arrow is evaluated
3) the variable is assigned the value of the expression
4) for each following variable steps 2 and 3 are performed sequentially.

If \( E \) is an expression, \( V \) is a variable, and \( L \) is the name of the statement, a labelled assignment statement has the form:

\[
L: \quad E \Rightarrow V
\]

Assignment Statement example

\[
A[I] + B \Rightarrow C[I](10 \Rightarrow 15).
\]

2.7.2 GO TO Statements and SWITCH Statements

Unconditional transfer of control statements are formed following the words \textit{GO TO} with a designational expression. Thus, the next statement to be executed will be one having
the value of the designational expression as its label.  
A **SWITCH** statement consists of a separate label by which it 
may be referenced; and names a group of alternative points in 
a program to which control may be transferred as the result 
of a single **GO TO** statement. The switch statement has the 
following form:

```
    ... **GO TO** L3. **GO TO** L2. **SWITCH** name : **GO TO** L1.
```

The selection of the actual point to which control is trans-
ferred depends on the value of the subscript expression of 
the switch designator in the **GO TO** statement. With increas-
ing value of the subscript expression an earlier label in 
the **SWITCH** statement is chosen for the transfer.

**GO TO** Statement example

```
    **GO TO** A.
    **GO TO** B[J].
```

**SWITCH** Statement example

```
    **GO TO** F. **GO TO** E. **GO TO** D. B:  **GO TO** C.
```

### 2.7.3 **FOR** Statements

The **FOR** statement facilitates writing an iterative 
operation one or more times. The variable which determines 
the number of executions is an index. The index takes on 
values beginning with a first limit and is modified by an in-
crement for each successive execution of the iterative opera-
tion. The execution of the **FOR** statement ends when a succes-
sive application of the increment would cause the index to 
pass beyond the second limit.
The **FOR** statement has the following form:

```
FOR index = first limit STEP increment UNTIL Second limit DO
  BEGIN statement S END
```

**FOR** statement example

```
FOR I = 0 STEP 1 UNTIL B DO
  BEGIN C[I] * D[I] \text{\(\rightarrow\)} E[I] END
```

2.7.4 **DO** Statements

A procedure or subroutine is a part of a program that is written only once but is to be executed at several points throughout the same program. A procedure is called for by a **DO** statement or procedure statement which effectively inserts the procedure body into the program taking the place of the **DO** statement. After the procedure has been executed the program continues with the next statement after the **DO** statement.

The format of the **DO** statement is:

```
DO Procedure Name,
```

The format of the **PROCEDURE** or subroutine is:

```
PROCEDURE Procedure Name:
  BEGIN Statement S_1, S_2, S_3, END
```

**DO** Statement example

```
DO INCREMENT,
```

**PROCEDURE** or Subroutine example

```
PROCEDURE INCREMENT:
  BEGIN J+1 = J, I+2 = I END
```
2.7.5 Conditional Statements

Conditional statements cause statements to be executed or skipped depending on the results of a Boolean expression or comparison. The conditional statement consists of a Boolean expression preceded by the word IF and followed by the word THEN, a "true part", and a "false part." Both "true" and "false parts" are unconditional statements. They are normally terminated by a semicolon, or by a period if the last statement was a GO TO statement. If the comparison is satisfied, the statement following THEN is executed after which control is transferred to the beginning of the next statement following the false part, unless the THEN statement terminates with a GO TO statement. If the comparison is not satisfied the ELSE statement is executed after which control is transferred to the beginning of the next statement unless a GO TO statement terminates the "false part."

Either "true" or "false parts" may be left vacuous by immediately terminating it with a semicolon.

The format of a conditional statement is:

```
IF Boolean Expression
   THEN unconditional statement, period or semicolon
   ELSE unconditional statement, period or semicolon
```

Conditional Statement example

```
IF A > B
   THEN A + B > C;
   ELSE GO TO D.
```
Chapter 3

INTERCOM 500

3.1 Intercom 500 computer

Intercom 500 is a programming system which is used on the Bendix G-15 digital computer. When Intercom 500 is stored in the G-15 memory, we essentially have an Intercom 500 digital computer. It is this computer that will be used as a simulation example. Included in the appendix is a BC NELIAC program simulating the Intercom 500 digital computer. The program has been tested and run successfully on the IBM 704 data processing system.

3.1.1 Computer Organization

The internal organization of the Intercom machine can be divided into five distinct functions: input, output, memory, arithmetic, and control. A diagram of the computer organization is shown in figure 1.

Three forms of input devices are available: paper tape, punched cards, and magnetic tape. The input information may consist of data or commands. Information may be put out in form of paper tape or on the typewriter.

The memory consists of 600 locations in which commands or data may be stored. Locations in the memory are specified by a four digit number called an "address." A command can be stored at, and executed from, any available address.

1. Intercom 500 card system
Intercom Computer Organization. Arrows represent direction of information flow.

Figure 1
Data also may be stored at any address.

The arithmetic section performs four operations - addition, subtraction, multiplication, and division. These operations are performed in a special register called the accumulator and this register can be addressed like any other location in memory.

The control section directs the operation of the computer. It consists of the current instruction register, location counter, and index registers. The Intercom machine has two important modes of operation: manual and automatic. During manual operation an instruction is read in directly from the input device to the current instruction register, where it is interpreted and executed. These instructions never appear in memory. The location counter has no meaning in the manual mode and the index registers operate as they do in the automatic mode.

In the automatic mode each instruction from memory has to be placed in the control section before it can be interpreted and executed. The current instruction register is the temporary storage in which each instruction is held while it is being interpreted after being brought from memory. Normally commands are obeyed in numerical sequence of their memory location. The location counter is given the address of the first command to be obeyed after which it keeps a running record of the location in memory of the instruction being executed. The index registers are available when automatic address modifi-
cation is desired. Each command which is used with an index register has its address modified by adding the contents of the index register to the address part of the instruction before the command is executed.

3.1.2 Command Structure

The machine instructions are in the form of numerically expressed commands which can be held in the internal memory. Each command is expressed by seven digits and sometimes an execution mark. The first digit of a command represents one of the ten index registers and may be left blank if no index is used. The next two digits specify the operation code which tells the machine what to do. The last four digits are termed the address part, and usually refer to a location in memory. If an instruction has an execution mark it will be interpreted and executed when it is read into the computer. The instruction will never appear in the internal memory and, therefore, not interfer in any way with the program.

3.1.3 Operation Codes

There are five major groups of operation codes available: arithmetic commands, transfer of control commands, input-output commands, index register commands, and special commands. The detail operation of these commands will be shown in the section on the simulation algorithm.

For a better understanding of the various modes in which the intercom machine will operate Figure 2 is given. Figure 2 is a block diagram showing the function of the input operation
Block diagram showing the function of the input operation codes and other important commands for transferring the machine between its various stages of operation. The "x" after an operation code indicates an execution mark.

Figure 2
codes and other important commands for transferring the machine between its various stages of operation. The 50 and 52 instructions will read commands and data, respectively, into memory. If these commands are executed while the machine is in the manual mode, information (commands or data) will be stored in memory beginning with the address specified in the read instruction. Information will be stored sequentially in memory until another command with an execution mark is interpreted. This command may be a 67 or 69 operation code which would transfer the machine to the manual or automatic mode, respectively. If the read commands (50 or 52) are executed while in the automatic mode, computation will halt and one word of information will be read into the computer and stored in the address specified in the read command; then computation will continue in the automatic mode. An 07 command will put the computer in a mode of operation for loading subroutines and a 61 command causes the machine to transfer to the selective print mode.

3.2 Algorithm for Simulation

Essentially, all intercom commands can be executed in any one of three modes: manual, automatic, or selective print mode. When the computer is in the manual mode, commands will be executed as they are read into the machine. In the automatic mode it is expected that the program is stored in the internal memory. The location counter is given the location of the first command in the program after which commands of
the program are automatically executed. The selective print mode is the same as the automatic mode except information concerning selected commands can be automatically typed out during computation. The computer is notified which command to type out by selectors provided in the program. The information typed will be the location of the command, the command itself, and the contents of the accumulator, if the contents of the accumulator is different than during the listing of a preceding command.

Figure 3 is a block diagram showing the basic operation of the BC NELIAC program which simulates the intercom 500 machine. The heart of the program is a large switch (EXECUTE) which is called as a procedure or subroutine (EXECUTE COMMAND) by any one of the three machine operating modes. This switch in turn calls the correct operation code, executes the command, and returns control to the original machine operating mode (except in the case of a command which changes operating modes).

The following simplified program written in BC NELIAC illustrates the operation of the algorithm for the manual and automatic modes and describes the function of each operation code. Read card is a procedure which inputs one word of information to the machine. Execute Command is a subroutine which transfers the program to the proper operation code subroutine.

3.3 Conclusions

Since the IBM 704 digital computer has a larger memory than the Bendix G-15, provisions have been made for a total memory size of 23,500 words in the BC NELIAC simulation on the
Block diagram showing the basic operation of the BC NELIAC program which simulates the Intercom 500 machine. The lines with arrows show the flow of the program while executing an operation code.

Figure 3
IBM 704 computer. A sample problem executed on the Intercom Simulation required 38 seconds running time as compared to 30 minutes on the Bendix G-15. To determine the advisability of developing a compiler for Intercom, a hand simulated compiled program of the sample problem was run on the IBM 704 and required 1.2 seconds. With speeds of 30 minutes for the G-15 and 1.2 seconds for the IBM 704 the figures would indicate the G-15 is not economical for operating Intercom problems. This fact is confirmed when considering the speed of operation of Intercom on an IBM 7090 computer. If an IBM 7090 is six times as fast as the IBM 704, a 5 hour Intercom problem on the G-15 would require 2 seconds on the IBM 7090.

The advantage of BC NELIAC as a source language is shown by the amount of time required to write and "debug" the Intercom 500 simulation program. It took five weeks for writing and "debugging" the program. The machine independent characteristics of BC NELIAC are indicated by the fact this simulation will be converted to the IBM 7090 in about three days. Most of the changes for the IBM 7090 will be in the input-output operations.
Simplified Program of Intercom 500 Manual and Automatic Modes

(COMMENT DECLARATION LIST)

A: ACCUMULATOR,
CR: COMMAND REGISTER: INDEX(7 > 10), OP CODE(0 > 6),
ADDRESS(11 > 18),
EA: EFFECTIVE ADDRESS,
IR: INDEX REGISTERS: W DIFFERENCE(10), W LIMIT(10),
W BASE(10), C BASE(10),
C DIFFERENCE(10), C LIMIT(10),
IRA: INDEX REGISTER ACCUMULATOR,
LC: LOCATION COUNTER,
M: MEMORY(23500),
MQ: MQ REGISTER,
MARK 1, MARK 2;

(COMMENT PROGRAM LOGIC)

MANUAL MODE: DO READ CARD, DO EXECUTE COMMAND,
GO TO MANUAL MODE.

(COMMENT READ CARD INPUTS ONE WORD OF
INFORMATION TO THE MACHINE.
EXECUTE COMMAND TRANSFERS PROGRAM
TO PROPER OP CODE SUBROUTINE)

AUTOMATIC MODE: STATE 1: M[LC] → CR,
IF INDEX ≠ 0
THEN ADDRESS + W BASE [INDEX]
+ C BASE [INDEX] → EA;
ELSE ADDRESS → EA;
STATE 2: DO EXECUTE COMMAND,
STATE 3: LC + 1 → LC, GO TO STATE 1.
## FUNCTION OF INTERCOM 500 OPERATION CODES

<table>
<thead>
<tr>
<th>Operation</th>
<th>Intercom 500 Operation Code</th>
<th>Symbolic Intercom Operation Code</th>
<th>Definition and/or Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ARITHMETIC COMMANDS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clear and Add</td>
<td>42</td>
<td>CLA:</td>
<td>{M[EA] \rightarrow A}</td>
</tr>
<tr>
<td>Clear and Subtract</td>
<td>40</td>
<td>CLS:</td>
<td>{- M[EA] \rightarrow A}</td>
</tr>
<tr>
<td>Clear and Add Absolute</td>
<td>45</td>
<td>CAB:</td>
<td>{</td>
</tr>
<tr>
<td>Store</td>
<td>49</td>
<td>STO:</td>
<td>{A &gt; M[EA]}</td>
</tr>
<tr>
<td>Add</td>
<td>43</td>
<td>FAD:</td>
<td>{A + M[EA] \rightarrow A}</td>
</tr>
<tr>
<td>Subtract</td>
<td>41</td>
<td>FSB:</td>
<td>{A - M[EA] \rightarrow A}</td>
</tr>
<tr>
<td>Multiply</td>
<td>44</td>
<td>FMP:</td>
<td>{A \times M[EA] \rightarrow A}</td>
</tr>
<tr>
<td>Divide</td>
<td>48</td>
<td>FDP:</td>
<td>{A / M[EA] \rightarrow A}</td>
</tr>
<tr>
<td>Inverse Divide</td>
<td>47</td>
<td>IFD:</td>
<td>{M[EA] / A \rightarrow A}</td>
</tr>
</tbody>
</table>
### FUNCTION OF INTERCOM 500 OPERATION CODES (cont.)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Intercom 500 Operation Code</th>
<th>Symbolic Intercom Operation Code</th>
<th>Definition and/or Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TRANSFER OF CONTROL COMMANDS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transfer</td>
<td>29</td>
<td>TRA:</td>
<td>{EA \rightarrow LC, \text{GO TO STATE 1.}},</td>
</tr>
<tr>
<td>Transfer on Non-negative</td>
<td>20</td>
<td>TNN:</td>
<td>{\text{IF } A \geq 0 \text{ THEN } \text{EA } \rightarrow \text{LC, GO TO STATE 1.;}},</td>
</tr>
<tr>
<td>Transfer on Negative</td>
<td>22</td>
<td>TRN:</td>
<td>{\text{IF } A &lt; 0 \text{ THEN } \text{EA } \rightarrow \text{LC, GO TO STATE 1.;}},</td>
</tr>
<tr>
<td>Transfer on Zero</td>
<td>23</td>
<td>TZE:</td>
<td>{\text{IF } A = 0 \text{ THEN } \text{EA } \rightarrow \text{LC, GO TO STATE 1.;}},</td>
</tr>
<tr>
<td>Transfer Mark Place 1</td>
<td>26</td>
<td>TMI:</td>
<td>{\text{LC } \rightarrow \text{MARK 1, EA } \rightarrow \text{LC, GO TO STATE 1.;}},</td>
</tr>
<tr>
<td>Return to Marked Place 1</td>
<td>16</td>
<td>RT1:</td>
<td>{\text{MARK 1 + 1 } \rightarrow \text{LC, GO TO STATE 1.;}},</td>
</tr>
<tr>
<td>Transfer Mark Place 2</td>
<td>28</td>
<td>TM2:</td>
<td>{\text{LC } \rightarrow \text{MARK 2, EA } \rightarrow \text{LC, GO TO STATE 1.;}},</td>
</tr>
<tr>
<td>Return to Marked Place 2</td>
<td>18</td>
<td>RT2:</td>
<td>{\text{MARK 2 + 1 } \rightarrow \text{LC, GO TO STATE 1.;}},</td>
</tr>
<tr>
<td>Transfer to Machine Subroutine</td>
<td>08</td>
<td>TSR:</td>
<td>{\text{DO MACHINE SUBROUTINE,}},</td>
</tr>
</tbody>
</table>
FUNCTION OF INTERCOM 500 OPERATION CODES (cont.)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Intercom 500 Operation Code</th>
<th>Symbolic Intercom Operation Code</th>
<th>Definition and/or Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX COMMANDS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assign Word Base</td>
<td>70</td>
<td>AWB:</td>
<td>{ADDR \to W BASE [INDEX]} ,</td>
</tr>
<tr>
<td>Assign Word Difference</td>
<td>71</td>
<td>AWD:</td>
<td>{ADDR \to W DIFFERENCE [INDEX]} ,</td>
</tr>
<tr>
<td>Assign Word Limit</td>
<td>72</td>
<td>AWL:</td>
<td>{ADDR \to W LIMIT [INDEX]} ,</td>
</tr>
<tr>
<td>Assign Channel Base</td>
<td>73</td>
<td>ACB:</td>
<td>{ADDR \to C BASE [INDEX]} ,</td>
</tr>
<tr>
<td>Assign Channel Difference</td>
<td>74</td>
<td>ACD:</td>
<td>{ADDR \to C DIFFERENCE [INDEX]} ,</td>
</tr>
<tr>
<td>Assign Channel Limit</td>
<td>75</td>
<td>ACL:</td>
<td>{ADDR \to C LIMIT [INDEX]} ,</td>
</tr>
<tr>
<td>Increment and Test 2</td>
<td>76</td>
<td>ITW:</td>
<td>{W BASE [INDEX]\ + N DIFFERENCE [INDEX]\ \to W BASE [INDEX],</td>
</tr>
<tr>
<td>Word Base</td>
<td></td>
<td></td>
<td>IF W BASE [INDEX]\ &lt; W LIMIT [INDEX]\ THEN |ADDR \to LC, GO TO STATE 1.;} ,</td>
</tr>
</tbody>
</table>
### FUNCTION OF INTERCOM 500 OPERATION CODES (cont.)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Intercom 500 Operation Code</th>
<th>Symbolic Intercom Operation Code</th>
<th>Definition and/or Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX COMMANDS (cont.)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Increment and Test Channel Base</td>
<td>77</td>
<td>ITC:</td>
<td>{ \text{C BASE} [\text{INDEX}] + \text{C DIFFERENCE} }</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[\text{INDEX}] \rightarrow \text{C BASE} }</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>{ IF \text{C BASE} [\text{INDEX}] \leq \text{C LIMIT} }</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[\text{INDEX}] \text{THEN} }</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADDR \rightarrow \text{LC}, \text{GO TO} }</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>STATE 1.;}</td>
</tr>
<tr>
<td>Set Index Accumulator</td>
<td>09</td>
<td>SIA:</td>
<td>{ \text{EA} \rightarrow \text{IRA} }</td>
</tr>
<tr>
<td>Clear and Add Index to IRA</td>
<td>78</td>
<td>CLI:</td>
<td>{ ADDR \times 11 + INDEX }</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\rightarrow \text{I}, \text{IR}[I] \rightarrow \text{IRA} }</td>
</tr>
<tr>
<td>Store Index from IRA</td>
<td>79</td>
<td>STI:</td>
<td>{ ADDR \times 11 + INDEX }</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\rightarrow \text{I}, \text{IRA} \rightarrow \text{IR}[I] }</td>
</tr>
</tbody>
</table>
### FUNCTION OF INTERCOM 500 OPERATION CODES (cont.)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Intercom 500 Operation Code</th>
<th>Symbolic Intercom Operation Code</th>
<th>Definition and/or Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INPUT OUTPUT COMMANDS</strong> - Following descriptions not in NELIAC form.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Originate Loading¹ Commands</td>
<td>50</td>
<td>ORG:</td>
<td>See sect. 3.1.3 for expl.</td>
</tr>
<tr>
<td>Read Command² Automatic</td>
<td>50</td>
<td>RCM:</td>
<td>See sect. 3.1.3 for expl.</td>
</tr>
<tr>
<td>Load Exponential Data¹</td>
<td>52</td>
<td>LDD:</td>
<td>See sect. 3.1.3 for expl.</td>
</tr>
<tr>
<td>Read Exponential Data² Automatic</td>
<td>52</td>
<td>RED:</td>
<td>See sect. 3.1.3 for expl.</td>
</tr>
<tr>
<td>Punch Binary Cards</td>
<td>39</td>
<td>PBC:</td>
<td>Binary cards punched from ADDR/100 * 100 to ADDR-1.</td>
</tr>
<tr>
<td>Read Binary Cards</td>
<td>55</td>
<td>RBC:</td>
<td>Absolute binary cards read into memory.</td>
</tr>
<tr>
<td>Position Typewriter, Tabs and carriage return</td>
<td>30</td>
<td>PTC:</td>
<td>ADDR/100 → No. of carriage returns ADDR-ADDR/100*100 → No. of tabs.</td>
</tr>
<tr>
<td>Write Literal and Tab</td>
<td>31</td>
<td>WLT:</td>
<td>EA printed, and typewriter tabbed.</td>
</tr>
<tr>
<td>Write Location Counter and Tab</td>
<td>06</td>
<td>WLC:</td>
<td>LC-1 printed, and typewriter tabbed.</td>
</tr>
<tr>
<td>Write Command and Tab</td>
<td>35</td>
<td>WCT:</td>
<td>M[EA] printed as command and typewriter tabbed.</td>
</tr>
<tr>
<td>Write Memory and Tab</td>
<td>37</td>
<td>WMT:</td>
<td>M[EA] printed in octa, and typewriter tabbed.</td>
</tr>
</tbody>
</table>
### Function of Intercom 500 Operation Codes (cont.)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Intercom 500 Operation Code</th>
<th>Symbolic Intercom Operation Code</th>
<th>Definition and/or Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INPUT OUTPUT COMMANDS (cont.)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write Floating Decimal and Tab</td>
<td>33</td>
<td>WFT:</td>
<td>M[EA] printed in floating decimal form, and typewriter tabbed.</td>
</tr>
<tr>
<td>Write Floating Decimal and Return Carriage</td>
<td>38</td>
<td>WFC:</td>
<td>M[EA] printed in floating decimal form, and typewriter carriage returned.</td>
</tr>
<tr>
<td>Write Exponential Data and Tab</td>
<td>32</td>
<td>WET:</td>
<td>M[EA] printed in exponential form, and typewriter tabbed.</td>
</tr>
<tr>
<td>Write Exponential Data and Return Carriage</td>
<td>34</td>
<td>WEC:</td>
<td>M[EA] printed in exponential form, and typewriter carriage returned.</td>
</tr>
</tbody>
</table>
### Function of Intercom 500 Operation Codes (cont.)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Intercom 500 Operation Code</th>
<th>Symbolic Intercom Operation Code</th>
<th>Definition and/or Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPECIAL COMMANDS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exit to Manual Mode</td>
<td>67</td>
<td>MAN:</td>
<td>{GO TO MANUAL MODE.}</td>
</tr>
<tr>
<td>Exit to Automatic Mode</td>
<td>69</td>
<td>AUT:</td>
<td>{EA → LC, GO TO STATE 1.}</td>
</tr>
<tr>
<td>No Operation</td>
<td>00</td>
<td>NOP:</td>
<td>{GO TO STATE 3.}</td>
</tr>
<tr>
<td>Ring Bell</td>
<td>63</td>
<td>BEL:</td>
<td>{DO RING BELL,}</td>
</tr>
<tr>
<td>Breakpoint Halt</td>
<td>68</td>
<td>BPH:</td>
<td>{GO TO MONITOR. ENDJOB.}</td>
</tr>
<tr>
<td>Load Subroutines(^1,3)</td>
<td>07</td>
<td>LSR:</td>
<td>{DO LOAD SUBROUTINES,}</td>
</tr>
<tr>
<td>Exit Loading Subroutines</td>
<td>00</td>
<td>ELS:</td>
<td>{DO EXIT LOAD SUBROUTINES,}</td>
</tr>
<tr>
<td>Initiate Selective Print(^1)</td>
<td>61</td>
<td>ISP:</td>
<td>Initiate Selective Print.</td>
</tr>
<tr>
<td>End Selective Print</td>
<td>62</td>
<td>ESP:</td>
<td>End Selective Print</td>
</tr>
</tbody>
</table>
### FUNCTION OF INTERCOM 500 OPERATION CODES (cont.)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Intercom 500 Operation Code</th>
<th>Symbolic Intercom Operation Code</th>
<th>Definition and/or Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit to Monitor Endjob</td>
<td>80</td>
<td>EJB:</td>
<td>{GO TO MONITOR ENDJOB.}</td>
</tr>
<tr>
<td>Read Clock</td>
<td>64</td>
<td>CLK:</td>
<td>{DO READ CLOCK,}</td>
</tr>
<tr>
<td>Load MQ</td>
<td>65</td>
<td>LDQ:</td>
<td>{M[EA] \rightarrow MQ}</td>
</tr>
<tr>
<td>Store MQ</td>
<td>66</td>
<td>STQ:</td>
<td>{M Q \rightarrow M[EA]}</td>
</tr>
</tbody>
</table>

### SPECIAL COMMANDS FOR IBM 704 INTERCOM

**Exponential Data**
- **EXD:** Used with exponential data

**Mask-Selector**
- **MSK:** Used with selectors for selective print.

**Equals**
- **EQU:** Assigns constant to symbol.

**Block Started by Symbol**
- **BSS:** Assigns block of storage to symbol.

**End Symbolic Program**
- **END:** Last card in symbolic program deck

**Blank**
- **---** Same as NOP
FUNCTION OF INTERCOM 500 OPERATION CODES (Footnotes)

1. Operation Code used only in the manual mode.
2. Operation Code(s) used only in the automatic mode.
3. See appendix
4. Component parts of an index register are symbolized by the contents of ADDR, as follows:

<table>
<thead>
<tr>
<th>IF ADDR</th>
<th>COMPONENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>W DIFFERENCE</td>
</tr>
<tr>
<td>1</td>
<td>W LIMIT</td>
</tr>
<tr>
<td>2</td>
<td>W BASE</td>
</tr>
<tr>
<td>3</td>
<td>C BASE</td>
</tr>
<tr>
<td>4</td>
<td>C DIFFERENCE</td>
</tr>
<tr>
<td>5</td>
<td>C LIMIT</td>
</tr>
</tbody>
</table>
4.1 Source Language

An Intercom 500 program is a sequence of seven digit commands which instructs the Intercom computer to perform a particular task. Symbolic Intercom has been developed as a source language more convenient for the programmer to use. There are approximately seventy different operation codes in Intercom 500. Writing programs using the numerical form for operation codes creates a complexity which is overcome by Symbolic Intercom. In Symbolic Intercom a symbolic code can be used for each of the operation codes, e.g. ADD for operation code 43. Since the numerical-type code does not have any of the mnemonic qualities of an alphabetic code nor does it provide a format that one may easily scan in order to see the meaning of a group of instructions, the symbolic form will necessarily result in faster and more accurate coding.

An additional function of Symbolic Intercom is overcoming the necessity of doing absolute coding. In absolute coding every word (command or data) in storage is assigned a location number used as a means of making references. This reference is made through the use of the address portion of the command. With absolute coding the programmer must determine the storage allocation in advance of coding. Since the storage requirements cannot be accurately anticipated, a re-design of the program may be necessary after its completion. If memory
space is limited, this re-design could cause rewriting of the program. Another problem occurs in absolute coding when an attempt is made to modify a program. Modifications usually entail insertions, deletions and re-arrangements of instructions. Every numerical reference made in the program to a location affected by the modifications must be changed so the program is still operative. This is a serious problem when there are several insertion and deletion areas. Every reference made must be tested to see how many of the different insertion and deletion areas affect it.

Symbolic-coding solves these problems of absolute coding because the basic method of referencing is changed. Instead of using an actual location number to indicate every reference made in the program, a location is given a name, or a symbol. This symbol has no numerical significance and no direct relationship to any particular storage-assignment scheme. The symbol is strictly a reference for the benefit of the programmer while writing his program.

A program which is given the name assembly program, defines where a symbolic program will sit in storage and what numerical location is assigned to each symbol. The assembly program also makes the translation between the symbolic operation code and the numeric operation code. Hence, if the input to the Intercom assembly program is Symbolic Intercom, the output will be Intercom 500.
In writing a symbolic program the following rules should be adhered to:

1. Every symbol is unique and independent of all other symbols.

2. If a symbol has been assigned to a particular location, all further references to this location may use the same symbol.

3. The locations of all instructions or data in a program having no reference made to them need no symbol assigned to them.

Included in the appendix is a BC NELIAC listing of the Symbolic Intercom assembly program. All Intercom 500 operation codes have a symbolic representation which are given in section 3.2. In addition, two psuedo operation codes have been added to the symbolic language - EQU and BSS. EQU allows the programmer to assign a constant to any symbol, and BSS provides for a block of storage to be assigned to a symbol. The accumulator can be addressed by the symbol ACC.

4.2 Algorithm

The Symbolic Intercom assembler is divided into two parts: first pass and second pass. In the first pass a storage cell, called the location counter, keeps track of the storage assignment of the current word in the program being assembled. The Intercom operation codes for read command or read data initialize the location counter. The location counter is increased by one for each word used by the program. The entire program
is examined sequentially during the first pass and any location with a symbolic name has this name put in a symbol table along with the current value of the location counter. Also, each symbolic operation code is converted to the appropriate Intercom 500 numerical "op code".

The second pass again examines the input sequentially and for each symbol used as an address in an instruction, replaces it with the appropriate location counter value from the symbol table. At the completion of the second pass, all symbolic commands have been converted to Intercom 500 instructions, and a copy of the symbol table, multiply defined and/or undefined symbols are printed out.
Appendix A

Transliteration rules

This appendix presents a summary of equivalences between the character set used with the hardware representation BC NELIAC on the IBM 704 digital computer and the BC NELIAC Reference Language. All word delimiters must be separated by blanks in the hardware representation.

<table>
<thead>
<tr>
<th>Character Operator</th>
<th>Hardware Representation</th>
<th>Reference Language Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Miscellaneous</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blank</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Operators</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replacement Operator</td>
<td>=</td>
<td>&gt;</td>
</tr>
<tr>
<td>Left Arrow</td>
<td>=</td>
<td>&lt;</td>
</tr>
<tr>
<td>Decimal Point</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td><strong>Punctuation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comma</td>
<td>,</td>
<td>,</td>
</tr>
<tr>
<td><strong>Operators</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Period</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>Semicolon</td>
<td>$</td>
<td>;</td>
</tr>
<tr>
<td><strong>Arithmetic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Subtract</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Multiply</td>
<td>*</td>
<td>X</td>
</tr>
<tr>
<td>Divide</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td><strong>Relational</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Less</td>
<td>LSS</td>
<td>&lt;</td>
</tr>
<tr>
<td>Less or Equal</td>
<td>LEQ</td>
<td>≤</td>
</tr>
<tr>
<td>Equal</td>
<td>EQU</td>
<td>=</td>
</tr>
<tr>
<td>Greater or Equal</td>
<td>GEQ</td>
<td>≥</td>
</tr>
<tr>
<td>Greater</td>
<td>GTR</td>
<td>&gt;</td>
</tr>
<tr>
<td>Not Equal</td>
<td>NEQ</td>
<td>≠</td>
</tr>
<tr>
<td><strong>Logical</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>And</td>
<td>AND</td>
<td>∧</td>
</tr>
<tr>
<td>OR</td>
<td>OR</td>
<td>∨</td>
</tr>
<tr>
<td><strong>Sequential</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GO TO</td>
<td>GO TO</td>
<td>GO TO</td>
</tr>
<tr>
<td>IF</td>
<td>IF</td>
<td>IF</td>
</tr>
<tr>
<td>FOR</td>
<td>FOR</td>
<td>FOR</td>
</tr>
</tbody>
</table>
Transliteration rules (cont.)

<table>
<thead>
<tr>
<th>Character Operator</th>
<th>Hardware Representation</th>
<th>Reference Language Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sequential</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operators (cont.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DO</td>
<td>DO</td>
<td>DO</td>
</tr>
<tr>
<td>THEN</td>
<td>THEN</td>
<td>THEN</td>
</tr>
<tr>
<td>ELSE</td>
<td>ELSE</td>
<td>ELSE</td>
</tr>
<tr>
<td><strong>Separator</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operators</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STEP</td>
<td>STEP</td>
<td>STEP</td>
</tr>
<tr>
<td>UNTIL</td>
<td>UNTIL</td>
<td>UNTIL</td>
</tr>
<tr>
<td>COLON</td>
<td>COLN</td>
<td>:</td>
</tr>
<tr>
<td>PERIOD</td>
<td></td>
<td>.</td>
</tr>
<tr>
<td>COMMA</td>
<td></td>
<td>,</td>
</tr>
<tr>
<td>SEMICOLON</td>
<td></td>
<td>;</td>
</tr>
<tr>
<td><strong>Bracket</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operators</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Left Parentheses</td>
<td>(</td>
<td>(</td>
</tr>
<tr>
<td>Right Parentheses</td>
<td>)</td>
<td>)</td>
</tr>
<tr>
<td>Left Bracket</td>
<td>LBK</td>
<td>[</td>
</tr>
<tr>
<td>Right Bracket</td>
<td>RBK</td>
<td>]</td>
</tr>
<tr>
<td>BEGIN, or Left Brace</td>
<td>BEGIN or LBR</td>
<td>BEGIN or{</td>
</tr>
<tr>
<td>END, or Right Brace</td>
<td>END or RBR</td>
<td>END or}</td>
</tr>
<tr>
<td><strong>Pseudo</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operators</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shift</td>
<td>EXP</td>
<td>EXP</td>
</tr>
<tr>
<td>Crutch Code</td>
<td>MCH</td>
<td>MCH</td>
</tr>
<tr>
<td>Octal</td>
<td>OCT</td>
<td>OCT</td>
</tr>
<tr>
<td>Alphabetic</td>
<td>A...Z</td>
<td>A....Z</td>
</tr>
<tr>
<td>Characters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Numeric Characters</td>
<td>0...9</td>
<td>0....9</td>
</tr>
</tbody>
</table>
OPERATION CODE LIMITATIONS

Due to hardware differences between the IBM 704 and Bendix G-15 the following operation codes will perform differently on the two machines:

Op Code 68: Breakpoint Halt: BPH

G-15: Computation is halted.
IBM 704: Transferred to Monitor Endjob.


G-15: Paper in the typewriter carriage is automatically positioned by the execution of CR carriage returns, followed by TB tabs.
CR is a two digit number ranging from 00 to 28.
TB is a two digit number ranging from 00 to 28.
IBM 704: Same as G-15 except tab settings are pre-set and allow a maximum of six columns of printout.

Op Code 37: Write Memory and Tab: WMT

G-15: The contents of location ADDR are typed out in hexadecimal form.
IBM 704: The contents of location ADDR are typed out in octal form.

Op Code 39: Punch Binary Cards: PBC

G-15: The contents of words 00 through ADDR-1 of the channel determined by the first two digits of ADDR are punched on paper tape.
IBM 704: Same as G-15 except cards are punched and no index registers will be punched on the cards.
Op Code 55: Read Binary Cards: RBC

G-15:  Punched tape, previously punched by the computer, is photo-electrically read and entered into the channel in the memory specified by the first two digits of ADDR. Information is entered in the channel beginning at word position 00 and ending with location ADDR-1.

IBM 704: Punch cards, previously punched by the computer, are stored in memory according to the absolute address on the column binary cards. ADDR has no significance.
Appendix C

INTERCOM CARD FORMAT

Intercom 500

Eighty column (numbered 1 - 80 from left to right)

IBM cards are used with one word of data or one command contained on a card. The card format for Intercom 500 will have the following form:

COLUMN: 1 - 63 64 66 68 69 70 72 74 76 78 80

COMMAND CARDS: COMMENT K O P A D D R S -

DATA CARDS: COMMENT E E D D D D D -

K = Index Register
OP = Operation Code
ADDRS = Address
EE = EXCESS Fifty Exponent
DDDDD = Datum

The card format for Symbolic Intercom is the following form:

COLUMN: 1 - 6 8 - 10 12 - 20 25 - 72

WORD
(DATA OR COMMAND): SYMBOL3 OP CODE VARIABLE FIELD4 COMMENT

1. A "minus" punched in column 78 on a data card indicates the data is negative.

2. A "minus" punched in column 80 on a command card indicates an execution mark.

3. An asterisk in column 1 of a command card indicates an execution mark.

4. Datum is indicated by an "op code" of EXD. The seven digit datum number is placed in the variable field (with a minus following the datum, if required.)
USE OF MACHINE LANGUAGE SUBROUTINES IN SYMBOLIC INTERCOM

Loading Subroutines

After executing the command LOAD SUBROUTINES, the following commands may be executed to store in memory the desired subroutines.

<table>
<thead>
<tr>
<th>SUBROUTINE</th>
<th>OP CODE</th>
<th>VARIABLE FIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fraction Selector</td>
<td>FRS</td>
<td>FRACTN</td>
</tr>
<tr>
<td>Square Root and Cube Root</td>
<td>SQT</td>
<td>SQTCUB</td>
</tr>
<tr>
<td>Log</td>
<td>LOG</td>
<td>LOG</td>
</tr>
<tr>
<td>Power</td>
<td>PWR</td>
<td>POWER</td>
</tr>
<tr>
<td>Sin and Cosine</td>
<td>TRG</td>
<td>TRIG</td>
</tr>
<tr>
<td>Arctangent</td>
<td>ART</td>
<td>ARCTAN</td>
</tr>
<tr>
<td>Hyperbolics</td>
<td>HYB</td>
<td>HYPBOL</td>
</tr>
<tr>
<td>Index Register Utilization</td>
<td>IRU</td>
<td>IRU</td>
</tr>
<tr>
<td>Selective Print</td>
<td>LSP</td>
<td>SELPRT</td>
</tr>
<tr>
<td>Clears Index Registers</td>
<td>CIR</td>
<td>XREGS</td>
</tr>
<tr>
<td>Clears Index Registers and Memory</td>
<td>CLM</td>
<td>MEMORY</td>
</tr>
</tbody>
</table>

Transfer to Machine Language Subroutines.

<table>
<thead>
<tr>
<th>SUBROUTINE</th>
<th>OP CODE</th>
<th>VARIABLE FIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selects Floating Decimal Fraction Length</td>
<td>TSR</td>
<td>DECPTO TO</td>
</tr>
<tr>
<td>Square Root of x</td>
<td>TSR</td>
<td>SQRT</td>
</tr>
<tr>
<td>Cube Root of x</td>
<td>TSR</td>
<td>CUBERT</td>
</tr>
<tr>
<td>Loge x</td>
<td>TSR</td>
<td>LOGE</td>
</tr>
</tbody>
</table>
Transfer to Machine Language Subroutines (cont.)

<table>
<thead>
<tr>
<th>SUBROUTINE</th>
<th>OP CODE</th>
<th>VARIABLE FIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log&lt;sub&gt;2&lt;/sub&gt; x</td>
<td>TSR</td>
<td>LOG2</td>
</tr>
<tr>
<td>Log&lt;sub&gt;10&lt;/sub&gt; x</td>
<td>TSR</td>
<td>LOG10</td>
</tr>
<tr>
<td>e&lt;sup&gt;x&lt;/sup&gt;</td>
<td>TSR</td>
<td>EXP</td>
</tr>
<tr>
<td>2&lt;sup&gt;x&lt;/sup&gt;</td>
<td>TSR</td>
<td>PWR2</td>
</tr>
<tr>
<td>10&lt;sup&gt;x&lt;/sup&gt;</td>
<td>TSR</td>
<td>PWR10</td>
</tr>
<tr>
<td>n&lt;sup&gt;m&lt;/sup&gt; (fixed point base-fixed point exponent)</td>
<td>TSR</td>
<td>EXP1</td>
</tr>
<tr>
<td>a&lt;sup&gt;m&lt;/sup&gt; (floating point base-fixed point exponent)</td>
<td>TSR</td>
<td>EXP2</td>
</tr>
<tr>
<td>a&lt;sup&gt;b&lt;/sup&gt; (floating point base-floating point exponent)</td>
<td>TSR</td>
<td>EXP3</td>
</tr>
<tr>
<td>Sin x (radians)</td>
<td>TSR</td>
<td>SIN</td>
</tr>
<tr>
<td>Sin x (degrees)</td>
<td>TSR</td>
<td>SIND</td>
</tr>
<tr>
<td>Cos x (radians)</td>
<td>TSR</td>
<td>COS</td>
</tr>
<tr>
<td>Cos x (degrees)</td>
<td>TSR</td>
<td>COSD</td>
</tr>
<tr>
<td>Arctan x (radians)</td>
<td>TSR</td>
<td>ATAN</td>
</tr>
<tr>
<td>Arctan x (degrees)</td>
<td>TSR</td>
<td>ATAND</td>
</tr>
<tr>
<td>Sinh x and Cosh x</td>
<td>TSR</td>
<td>SINH</td>
</tr>
<tr>
<td>Tanh x</td>
<td>TSR</td>
<td>TANH</td>
</tr>
<tr>
<td>Fix Floating Point Number</td>
<td>TSR</td>
<td>FIX</td>
</tr>
<tr>
<td>Float Fixed Point Number</td>
<td>TSR</td>
<td>FLOAT</td>
</tr>
</tbody>
</table>
Appendix E

SYMBOLIC INTERCOM 500 ASSEMBLER

IR CLN LBR INDEX REGISTER (30-35) RBR.
U CLN LBR OF TEN (100-251) RBR.
V CLN LBR OF UNIT (36-351) RBR.
ADDR 1 CLN LBR CH. TEN (36-351) RBR.
ADDR 2 CLN LBR CH. UNIT (36-351) RBR.
ADDR 3 CLN LBR NO. UNIT (36-351) RBR.
ADDR 4 CLN LBR NO. UNIT (36-351) RBR.
SIGN VALUE CLN LBR DATA SIGN (30-351) RBR.
X CLN LBR X MARK (30-351) RBR.

(COMMENT - ARRAY DIMENSION LIST)
COMMENT A (1900).
COMMENT B (1900).
COMMENT C (1900).
COMMENT D (1900).
COMMENT E (1900).
COMMENT F (1900).
COMMENT G (1900).
COMMENT H (1900).
EX MARK (1900).
OPERATION (1900).
SYMBOL FIELD (1900).
MULTIPLY DEFINED SYMBOLS (126).
UNDEFINED SYMBOLS (126).

(COMMENT - LOGICAL VARIABLE DIMENSION LIST)
DCD NAME (26).
COMMAND OP CODE (46) (46).
DATA, LAST.
NAME (46).
SYMBOL (22).
VARIABLE FIELD A (47) (2200).
VARIABLE FIELD B (48) (2200).

(COMMENT - CONSTANT DIMENSION LIST)
ALL BLANK = OCT = 2000 (3000).
ASTERISK = OCT = 1400 (3000).
DCD 9 = OCT = 11.
DIMENSIONING

FLOWCHART NUMBER 00001

$START SIN CLN SYMBOLIC INTERCOM 500 CLN0000
MCH 0772000 OCT 205, IOH PRINT (50,0), FORTY ONE = N,
ZERO = I, GO TO FIRST PASS..

FLOWCHART NUMBER 00002

(COMMENT SIMPLE VARIABLE DIMENSION LIST)

BCD DIGIT (4),
CALL PUNCH,
CHARACTER,
CM1, CM2, CM3, CM4, CM5, CM6, CM7, CM8,
DIGIT,
LOCATION COUNTER,
NUMB,
PROGRAM LENGTH,
SYM,
SYMBOL TABLE LENGTH,

IR CLN LBR INDEX REGISTER(30=35) RBR = OCT 1
O CLN LBR OP TEN(30=35) RBR = OCT 2
P CLN LBR OP UNIT(30=35) RBR = OCT 3
 ADDR 1 CLN LBR CH TEN(30=35) RBR = OCT 4
 ADDR 2 CLN LBR CH UNIT(30=35) RBR = OCT 5
 ADDR 3 CLN LBR WD TEN(30=35) RBR = OCT 6
 ADDR 4 CLN LBR WD UNIT(30=35) RBR = OCT 7
SIGN VALUE CLN LBR DATA SIGN(30=35) RBR = OCT 8
X CLN LBR X MARK(30=35) RBR = OCT 9

(COMMENT ARRAY DIMENSION LIST)

COMMENT A (900),
COMMENT B (900),
COMMENT C (900),
COMMENT D (900),
COMMENT E (900),
COMMENT F (900),
COMMENT G (900),
COMMENT H (900),
EX MARK (900),
OPERATION (900),
SYMBOL FIELD (900),
MULTIPLY DEFINED SYMBOLS (26),
UNDEFINED SYMBOLS (26),

(COMMENT LOGICAL VARIABLE DIMENSION LIST)

BCD NUMB (*6),
COMMAND OP CODE (*6) (900),
DATA (*6),
NAME (*6),
SYMOL (*6),
VARIABLE FIELD A (*6) (900),
VARIABLE FIELD B (*6) (900),

(COMMENT CONSTANT DIMENSION LIST)

ALL BLANK = OCT -206060606060,
ASTERISK = OCT -146060606060,
BCD 9 = OCT 11,
Operation Table (74) = Oct 234321606060,
                (Comment CLA = 42)
                Oct -226346606060, (Comment STO = 49)
                Oct 262124606060, (Comment FAD = 43)
                Oct 266222606060, (Comment FSB = 41)
                Oct 264447606060, (Comment FMP = 44)
                Oct 262447606060, (Comment FDP = 48)
                Oct -235121606060, (Comment TBA = 29)
                Oct -237125606060, (Comment TZE = 23)
                Oct 234362606060, (Comment CLS = 40)
                Oct 312624606060, (Comment IFD = 47)
                Oct -234545606060, (Comment TNN = 20)
                Oct -235145606060, (Comment TRN = 22)
                Oct -054647606060, (Comment NOP = 00)
                Oct -234401606060, (Comment TM1 = 26)
                Oct -116301606060, (Comment RT1 = 16)
                Oct -234402606060, (Comment TM2 = 28)
                Oct -116302606060, (Comment RT2 = 18)
                Oct -236251606060, (Comment TSR = 08)
                Oct 256724606060, (Comment EXD = 91)
                Oct 316366606060, (Comment ITW = 76)
                Oct -262563606060, (Comment WET = 32)
                Oct -262523606060, (Comment WEQ = 34)
                Oct -262663606060, (Comment WFT = 33)
                Oct -262623606060, (Comment WFG = 38)
                Oct 216622606060, (Comment AWB = 70)
                Oct -112524606060, (Comment RED = 52)
                Oct -112344606060, (Comment RCM = 50)
                Oct 216463606060, (Comment AUT = 69)
                Oct -042145606060, (Comment MAN = 67)
                Oct 216624606060, (Comment AWD = 71)
                Oct 216643606060, (Comment AWL = 72)
                Oct -076323606060, (Comment PTC = 30)
                Oct 232122606060, (Comment CAB = 45)
                Oct 212322606060, (Comment ACB = 73)
                Oct 212324606060, (Comment ACD = 74)
                Oct 212343606060, (Comment ACL = 75)
                Oct 316323606060, (Comment ITC = 77)
                Oct 254122606060, (Comment EJB = 80)
                Oct -264323606060, (Comment WLC = 06)
                Oct -262363606060, (Comment WCT = 35)
                Oct 316247606060, (Comment ISP = 61)
                Oct -046242606060, (Comment MSK = 90)
                Oct 256247606060, (Comment ESP = 62)
<table>
<thead>
<tr>
<th>Oct</th>
<th>Comment</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCT -264363606060</td>
<td>WLT = 31</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT -072223606060</td>
<td>PBC = 39</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT -1122360606060</td>
<td>RBC = 55</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 234331606060</td>
<td>CLI = 78</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT -226331606060</td>
<td>STI = 79</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT -223121606060</td>
<td>SIA = 09</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 224323606060</td>
<td>BLC = 81</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 234342606060</td>
<td>CLK = 64</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 2254360606060</td>
<td>WMT = 37</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 224730606060</td>
<td>LSR = 07</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 222543606060</td>
<td>BEL = 63</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT -032450606060</td>
<td>BPH = 68</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT -076651606060</td>
<td>LOC = 65</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT -226350606060</td>
<td>STQ = 66</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT -264463606060</td>
<td>WMT = 37</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT -036251606060</td>
<td>LSP = 00</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 233151606060</td>
<td>CR = 00</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 254362606060</td>
<td>ELS = 00</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT -206060606060</td>
<td>BLANK = 00</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 226262606060</td>
<td>BSS = 82</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 255064606060</td>
<td>EQU = 83</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT -065127606060</td>
<td>ORG = 50</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT -034627606060</td>
<td>LOG = 03</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT -076651606060</td>
<td>PWR = 04</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT -235127606060</td>
<td>TRG = 05</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 215163606060</td>
<td>ART = 06</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 307022606060</td>
<td>HYB = 07</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 315164606060</td>
<td>IRU = 12</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 254362606060</td>
<td>ELS = 00</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT -034627606060</td>
<td>BLANK = 00</td>
<td>(COMMENT)</td>
</tr>
</tbody>
</table>

**Operation Code (6) (74)**

<table>
<thead>
<tr>
<th>Oct</th>
<th>Comment</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCT 040200000000</td>
<td>CLA = 42</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 040900000000</td>
<td>STO = 49</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 040300000000</td>
<td>FAD = 43</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 040100000000</td>
<td>FS = 41</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 040400000000</td>
<td>FMP = 44</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 040800000000</td>
<td>FDP = 48</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 020900000000</td>
<td>TRA = 29</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 020300000000</td>
<td>TZE = 23</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 040000000000</td>
<td>CLS = 40</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 040700000000</td>
<td>IFD = 47</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 020000000000</td>
<td>TNN = 20</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 020200000000</td>
<td>TRN = 22</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 000000000000</td>
<td>ND = 00</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 020600000000</td>
<td>TM1 = 26</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 010600000000</td>
<td>RT1 = 16</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 010800000000</td>
<td>TM2 = 28</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 000800000000</td>
<td>TSR = 08</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 090100000000</td>
<td>EXD = 91</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 070600000000</td>
<td>ITW = 76</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 030200000000</td>
<td>WET = 32</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 030400000000</td>
<td>WE = 34</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 030300000000</td>
<td>WFT = 33</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 030800000000</td>
<td>WFC = 38</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 070000000000</td>
<td>AWB = 70</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 050200000000</td>
<td>RED = 52</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 050000000000</td>
<td>RCM = 50</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 060900000000</td>
<td>AUT = 69</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 060700000000</td>
<td>MAN = 67</td>
<td>(COMMENT)</td>
</tr>
</tbody>
</table>

**Symbol Table (41)**

<table>
<thead>
<tr>
<th>Oct</th>
<th>Comment</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCT 040900000000</td>
<td>CLA = 42</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 040900000000</td>
<td>STO = 49</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 040300000000</td>
<td>FAD = 43</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 040100000000</td>
<td>FS = 41</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 040400000000</td>
<td>FMP = 44</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 040800000000</td>
<td>FDP = 48</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 020900000000</td>
<td>TRA = 29</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 020300000000</td>
<td>TZE = 23</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 040000000000</td>
<td>CLS = 40</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 040700000000</td>
<td>IFD = 47</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 020000000000</td>
<td>TNN = 20</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 020200000000</td>
<td>TRN = 22</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 000000000000</td>
<td>ND = 00</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 020600000000</td>
<td>TM1 = 26</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 010600000000</td>
<td>RT1 = 16</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 010800000000</td>
<td>TM2 = 28</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 000800000000</td>
<td>TSR = 08</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 090100000000</td>
<td>EXD = 91</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 070600000000</td>
<td>ITW = 76</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 030200000000</td>
<td>WET = 32</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 030400000000</td>
<td>WE = 34</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 030300000000</td>
<td>WFT = 33</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 030800000000</td>
<td>WFC = 38</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 070000000000</td>
<td>AWB = 70</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 050200000000</td>
<td>RED = 52</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 050000000000</td>
<td>RCM = 50</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 060900000000</td>
<td>AUT = 69</td>
<td>(COMMENT)</td>
</tr>
<tr>
<td>OCT 060700000000</td>
<td>MAN = 67</td>
<td>(COMMENT)</td>
</tr>
</tbody>
</table>
ASSEMBLER FLOWCHART NUMBER 00003

$DIMENSIONING CLN ..

SYMBOL TABLE VALUE (41) = OCT 000000000000, OCT 000000010000, OCT 000000020000, OCT 000000030000, OCT 000000040000, OCT 000000050000, OCT 000000060000, OCT 000000070000, OCT 000000080000, OCT 000000090000, OCT 0000000A0000, OCT 0000000B0000, OCT 0000000C0000, OCT 0000000D0000, OCT 0000000E0000, OCT 0000000F0000.

CHECK OCT 256747016060, OCT 256747026060, OCT 256747036060, OCT 256747046060, OCT 256747056060, OCT 256747066060, OCT 256747076060, OCT 256747086060, OCT 256747096060, OCT 2567470A6060, OCT 2567470B6060, OCT 2567470C6060, OCT 2567470D6060, OCT 2567470E6060, OCT 2567470F6060.

$FIRST PASS CLN

READ PROGRAM CLN ADDRESS, NUMB = LOCATION COUNTER, LDD OP I + ONE = I, IOH READ (51,0, SYMBOL FIELD, OPERATION, VARIABLE FIELD A, CM1, CM2, CM3, CM4, CM5, CM6, CM7, CM8), IF OPERATION EQU EENO THEN I = PROGRAM LENGTH, VARIABLE FIELD A = CALL PUNCH(*1), N - ONE = SYMBOL TABLE LENGTH, CONVERT SYMBOL TABLE TO BCD.$ FOR J = 1 TO 73 DO BEG IF OPERATION EQU OPERATION TABLE LBK J RBK THEN GO TO CONVERT SYMBOL TABLE TO BCD.$ END, THE THOUSAND LBK J RBK IOH PRINT (52,0, OPERATION), GO TO ENDJ6. LBK J RBK = DIGIT.
CHECK PSEUDO OP CLN

IF J GTR LAST OP CODE
THEN GO TO PSEUDO OP CLN
OPERATION CODE LBK J RBK = COMMAND OP CODE LBK I RBK
STORE CARD CLN

SYMBOL FIELD = SYMBOL FIELD LBK I RBK, OPERATION
= OPERATION LBK I RBK, VARIABLE FIELD A
= VARIABLE FIELD A LBK I RBK, VARIABLE FIELD B
SECOND = VARIABLE FIELD B LBK I RBK, CM1 = COMMENT A LBK I RBK
CM2 = COMMENT B LBK I RBK, CM3 = COMMENT C LBK I RBK
CM4 = COMMENT D LBK I RBK, CM5 = COMMENT E LBK I RBK
CM6 = COMMENT F LBK I RBK, CM7 = COMMENT G LBK I RBK
CM8 = COMMENT H LBK I RBK, IF SYMBOL FIELD EQU ASTERISK
THEN MINUS SIGN = EX MARK LBK I RBK, GO TO READ PROGRAM.
ELSE ALL BLANK = EX MARK LBK I RBK
IF SYMBOL FIELD NEQ ALL BLANK
THEN DO ENTER SYMBOL IN TABLE, $$
LOCATION COUNTER + ONE = LOCATION COUNTER, GO TO READ PROGRAM.

GO TO LDD.  GO TO ORG.  GO TO EQUAL.
PSEUDO OPERATION CLN  GO TO BSS.

BSS CLN
LDD OP = COMMAND OP CODE LBK I RBK, IF SYMBOL FIELD
= SYMBOL FIELD LBK I RBK NEQ ALL BLANK
THEN DO ENTER SYMBOL IN TABLE, $$
DO CONVERT ADDRESS, LOCATION COUNTER + NUMB - ONE
= LOCATION COUNTER = NUMB, VARIABLE FIELD A
VAR TABLE LBK I RBK = LOWER TABLE LBK K RBK
CONVERT LOCATION COUNTER TO BCD CLN
FOR J EQU 0 STEP 1 UNTIL 2 DO
BEGIN NUMB /ONE THOUSAND LBK J RBK = BCD DIGIT LBK J RBK
* ONE THOUSAND LBK J RBK = DIGIT, NUMB - DIGIT
= NUMB, END,
NUMB = BCD DIGIT LBK 3 RBK, FOR J EQU 1 STEP 1 UNTIL 3 DO
BEGIN BCD DIGIT(#6) ++ BCD DIGIT LBK J RBK (#6)
= BCD DIGIT(#6), END,
BCD DIGIT = 2 * 12 + PART BLANK = VARIABLE FIELD A LBK I RBK
OPERATION = OPERATION LBK I RBK, CM1 = COMMENT A LBK I RBK
CM2 = COMMENT B LBK I RBK, CM3 = COMMENT C LBK I RBK
CM4 = COMMENT D LBK I RBK, CM5 = COMMENT E LBK I RBK
CM6 = COMMENT F LBK I RBK, CM7 = COMMENT G LBK I RBK
CM8 = COMMENT H LBK I RBK, MINUS SIGN = EX MARK LBK I RBK
GO TO READ PROGRAM.

EQUAL CLN

I0H PRINT (60,0, SYMBOL FIELD, OPERATION, VARIABLE FIELD A,
CM1, CM2, CM3, CM4, CM5, CM6, CM7, CM8),
DO ENTER SYMBOL IN TABLE, DO CONVERT ADDRESS,
NUMB = SYMBOL TABLE VALUE LBK N-1 RBK, I - ONE = I,
GO TO READ PROGRAM.

LDD CLN
DO CONVERT ADDRESS, NUMB = LOCATION COUNTER, LDD OP
= COMMAND OP CODE LBK I RBK, GO TO STORE CARD.

ORG CLN
DO CONVERT ADDRESS, NUMB = LOCATION COUNTER, ORG OP
= COMMAND OP CODE LBK I RBK, GO TO STORE CARD.

CONVERT SYMBOL TABLE VALUE TO BCD CLN
FOR I EQU 41 STEP 1 UNTIL SYMBOL TABLE LENGTH DO
BEGIN FOR J EQU ZERO STEP 1 UNTIL 2 DO
BEGIN SYMBOL TABLE VALUE LBK I RBK / ONE THOUSAND LBK J RBK
= BCD DIGIT LBK J RBK * ONE THOUSAND LBK J RBK = DIGIT,
WRITE OUTPUT
SYMBOL TABLE VALUE LBK I RBK - DIGIT
SYMBOL TABLE VALUE LBK I RBK - END
FOR J EQU 0 STEP 1 UNTIL 3 DO TABLE FIELD BCD
BEGIN BCD DIGIT(*6) ++ BCD DIGIT LBK J RBK (*6)
COMMENT = BCD DIGIT (*6), END,
GOTLBK I RBK = CM4,
FOR BCD DIGIT * 2 EXP 12 = SYMBOL TABLE VALUE LBK I RBK, END,
ZERO = I, CONNECT LBK I RBK = CM8,
SECOND PASS CLN I + ONE = I, IF I EQU PROGRAM LENGTH IM, CM8, CM4, CH7, CM8, IR, O, P,
THEN IOH PRINT (55,0), GO TO WRITE TABLES ON TAPE 9.$
IF COMMAND OP CODE LBK I RBK GTR MAXIMUM COMMAND CODE
THEN VARIABLE FIELD A LBK I RBK = DATA,
FOR J EQU 0 STEP 1 UNTIL 5 DO I BEGIN *DATA = INDEX REGISTER LBK J RBK, DATA += DATA END,
ALL VARIABLE FIELD B LBK I RBK = DATA, FOR J EQU 0 STEP 1 UNTIL 2
WRITE J DO BEGIN *DATA = WD UNIT LBK J RBK, DATA += DATA, END,
FOR GO TO WRITE OUTPUT TAPES.$
OUTPUT COMMAND CLN TABLE VALUE LBK I RBK - SYMB
*COMMAND OP CODE LBK I RBK = OP TEN, COMMAND OP CODE LBK I RBK
++ COMMAND OP CODE LBK I RBK, * COMMAND OP CODE LBK I RBK
= OP UNIT, EX MARK LBK I RBK = X,
VARIABLE FIELD A LBK I RBK = NAME = SYMB[*1], UNTIL L-1 DO
IF SYMB EQU ALL BLANK DEFINED SYMBOL LBK J RBK = SYMB,
THEN ZERO = NAME $$, Go TO OUTPUT BCD
IF *NAME GTR BCD 9
THEN BEGIN DO BUILD SYMBOL,
FOR K EQU 0 STEP 1 UNTIL N-1 DO
BEGIN IF SYMB EQU SYMBOL TABLE LBK K RBK
EXIT ASSEMBLER CLN THEN SYMBOL TABLE VALUE LBK K RBK (*1) = BCD NUMB,
RCH 0770000 OCT GO TO OUTPUT BCD VARIABLE FIELD.$ END,
IF CALSYM = UNDEFINED SYMBOLS LBK M RBK, M + ONE = M,
THEN ZERO = BCD NUMB, END $$ WRITE I-5 OR I-9 BCD CARD IMAGES
ELSE BEGIN RIGHT ADJUST CLN
START INTERCOM 500 INTERPRETER
IF NAME(0=5) EQU SHORT BLANK
ASSEMBLER SUBROUTINES THEN NAME / 2 EXP 6 = NAME, GO TO RIGHT ADJUST.$
IF NAME(6=11) NEQ COMMA
FLOWCHART NUMBER 00 THEN NAME * 2 EXP 12 = NAME $$
FOR J EQU 0 STEP 1 UNTIL 3 DO
PROCEDURE BEGIN *NAME = CH TEN LBK J RBK, NAME += NAME, END
BEGIN SYMP = NAME = CHARACTER EQM COMMAND 0 STEP 1 UNTIL 9 DO
BEGIN THEN NAME += NAME, *NAME = INDEX REGISTER,
IF CHAR = Go TO WRITE OUTPUT TAPES,
ELSE ZERO = INDEX REGISTER, GO TO WRITE OUTPUT TAPES.$
OUTPUT BCD VARIABLE FIELD CLN = SYMBOL $-
FOR K EQU 0 STEP 1 UNTIL 3 DO
BEGIN BCD NUMB = CH TEN LBK K RBK,
ADJUST CHAR = BCD NUMB += BCD NUMB, END
OUTPUT INDEX CLN
IF = NEQ FIVE SHORT BLANK = SYMBOL
THEN Go TO CHECK INDEX.$
SYMP = CHARACTER EQM COMA N RBK (*1), LOCATION COUNTER
= SYMP THEN *VARIABLE FIELD B LBK I RBK = INDEX REGISTER, L N-1 DO
BEGIN Go TO WRITE OUTPUT TAPES.$
EQUAL SYMBOL TABLE LBK N RBK
IF *VARIABLE FIELD B LBK I RBK EQM COMA
THEN VARIABLE FIELD B LBK I RBK = NAME, NAME += NAME,
*NAME = INDEX REGISTER, GO TO WRITE OUTPUT TAPES
EXIT ELSE ZERO = IR, Go TO WRITE OUTPUT TAPES.$
CHECK INDEX CLN END = SYMBOL END
IF CHARACTER NEQ COMA
PROCEDURE CONV THEN ZERO = IR, Go TO WRITE OUTPUT TAPES.$
BEGIN VARNAME += NAME, *NAME = INDEX REGISTER, GTR BCD 9
WRITE OUTPUT TAPES CLN

WRITE TAPE 5 CLN

WRITE TAPE 9 CLN

WRITE TABLES ON TAPE 9 CLN

ASSEMBLER SUBROUTINES

FLOWCHART NUMBER 00004

$PROCEDURE ENTER SYMBOL IN TABLE CLN
BEGIN SYMBOL FIELD(#) = NAME, FOR J EQU O STEP 1 UNTIL 5 DO
BEGIN *NAME = CHARACTER,
IF CHARACTER EQU SHORT BLANK
THEN GO TO EXIT.
ELSE SYMBOL ++ CHARACTER = SYMBOL $ $;
EXIT CLN
NAME += NAME, END ,
ADJUST CHARACTERS LEFT CLN
IF *SYMBOL EQU ZERO
THEN SYMBOL ++ SHORT BLANK = SYMBOL,
GO TO ADJUST CHARACTERS LEFT.$
SYMBOL = SYMBOL TABLE LBK N RBK (#), LOCATION COUNTER
= SYMBOL TABLE VALUE LBK N RBK , FOR J EQU 0 STEP 1 UNTIL N-1 DO
BEGIN IF SYMBOL TABLE LBK J RBK EQU SYMBOL TABLE LBK N RBK
THEN SYMBOL TABLE LBK N RBK
= MULTIPLY DEFINED SYMBOLS LBK L RBK , L + ONE = L,
GO TO EXIT SYMBOL TABLE ENTRY.$ END ,
EXIT SYMBOL TABLE ENTRY CLN
N + ONE = N, ZERO = SYMBOL, END ,

PROCEDURE CONVERT ADDRESS CLN
BEGIN VARIABLE FIELD A = NAME, IF *NAME = NUMB GTR BCD 9
ASSEMBLER FOR THEN BEGIN DO BUILD SYMBOL, FOR K EQU N-1 STEP -1 UNTIL 0 DO
BEGIN IF SYMB EQU SYMBOL TABLE LBK K RBK THEN SYMBOL TABLE VALUE LBK K RBK = NUMB,
GO TO EXIT CONVERT ADDRESS.$ END,
CONTROL ZERO = NUMB, END $ 501 ELSE BEGIN CONVERT BCD NUMBER TO BINARY CLN NAME += NAME, IF *NAME = BCD NUMB EQU COMMA THEN GO TO EXIT CONVERT ADDRESS.$ IF BCD NUMB NEQ SHORT BLANK THEN BCD NUMB + NUMB * TEN = NUMB, GO TO CONVERT BCD NUMBER TO BINARY.$ END $ EXIT CONVERT ADDRESS CLN NAME = CHARACTER, IF CHARACTER EQU SHORT BLANK THEN GO TO EXIT BUILD SYMBOL.$ IF CHARACTER EQU COMMA THEN GO TO EXIT BUILD SYMBOL.$ SYMB ++ CHARACTER = SYMB, NAME += NAME, END,
EXIT BUILD SYMBOL CLN IF *SYMB EQU ZERO THEN SYMB ++ SHORT BLANK = SYMB,
GO TO EXIT BUILD SYMBOL.$ SYMB = SYMB[*1], ZERO = SYMB, END...
ASSEMBLER FORMATS

FLOWCHART NUMBER 00005

CONTROL
50(14H 1SYMBOLIC MODE/)  
51(C6,1XC3,1XC6,C3,4X8C6)  
52(27H  ILLEGAL OPERATION CODE = C3)  
53(8C6,14X9(1XC1))  
54(3X9C1,3XC6,1XC3,1XC6,C6,5X8C6)  
55(/ / / 13H SYMBOL TABLE/X)  
56(3XC6,5XC4)  
57(/ / / 25H MULTIPLY DEFINED SYMBOLS/X)  
58(3XC6)  
59(/ / / 18H UNDEFINED SYMBOLS/X)  
60(15XC6,1XC3,1XC6,11X8C6)
Appendix F

BC NELIAC SIMULATION OF INTERCOM 500
FLOWCHART NUMBER 00001

$START
INTERCOM 500 INTERPRETER CLN
FORMAT ADDRESS = PRINTOUT LBK 32766 RBK, FIRST PRINT VARIABLE = L,
FIRST FORMAT = M, GO TO PERMIT MANUAL OPERATION.

FLOWCHART NUMBER 00002

(COMMENT VARIABLE DIMENSION LIST)

ACCUMULATOR = 0.0*0,
ADDRESS (7),
ADDRESS SEPARATION,
BCD NUMBER (4),
CHL (7),
COL 64 CLN LBR COLUMN 64(30=33), COLUMN K(30=35) RBR,
COL 66 CLN LBR COLUMN 66(30=33), COLUMN O(30=35) RBR,
COL 68 CLN LBR COLUMN 68(30=33), COLUMN P(30=35) RBR,
COL 69 CLN LBR COLUMN 69(30=33), COLUMN C(30=35) RBR,
COL 70 CLN LBR COLUMN 70(30=33), COLUMN H(30=35) RBR,
COL 72 CLN LBR COLUMN 72(30=33), COLUMN L(30=35) RBR,
COL 74 CLN LBR COLUMN 74(30=33), COLUMN W(30=35) RBR,
COL 76 CLN LBR COLUMN 76(30=33), COLUMN D(30=35) RBR,
COMMAND A,
COMMAND B,
COMMENT1 CLN C1, COMMENT2 CLN C2, COMMENT3 CLN C3,
COMMENT4 CLN C4, COMMENT5 CLN C5, COMMENT6 CLN C6,
COMMENT7 CLN C7, COMMENT8 CLN C8, COMMENT9 CLN C9,
COMMENT10 CLN C10, COMMENT11 CLN C11,
DATUM (7) = 0.0*0,
EMPTY WORD CLN EW,
EXECUTE,
EXPONENT,
FIRST SELECTOR A,
FIRST SELECTOR B,
FRACTION SELECTOR CHANNEL CLN FRAC SEL CHANNEL,
INDEX (7),
INDEX REGISTER ACCUMULATOR CLN IRA,
INDEX REGISTER COMPONENT ADDRESS CLN IR COMP ADDRESS,
INDEX REGISTER UTILIZATION CHANNEL CLN IRU CHANNEL,
J TEMPORARY STORAGE CLN JTS,
LAST WORD,
LOCATION,
LOG CHANNEL,
MARK PLACE 1,
MARK PLACE 2,
MEMORY (*128) * OCT 65000, CELL * OCT 65000,
MODE, 1,
MQ = 2,
NUMBER,
OP CODE (7),
PAST ACCUMULATOR,
POWER CHANNEL,
PROGRAM STORAGE,
SECOND SELECTOR,
SECOND SELECTOR A,
SECOND SELECTOR B,
SELECTOR = 1,
SIGN, FIVE = 55,
STORAGE, FIVE = 65,
SUBROUTINE, FIVE = 71,
TAB = 1,
TABL (7),
WD (7),
W DIFFERENCE (11),
W LIMIT (11),
W BASE (11),
C BASE (11),
C DIFFERENCE (11),
C LIMIT (11),

(COMMENT CONSTANT DIMENSION LIST)

ADDR INCREMENT = OCT 13000,
AUTOMATIC = 1,
BLANK = OCT -206060606060,
BCD 100 = OCT 010000,
CARRIAGE RETURN FORMAT CLN LBR CR (24=35) RBR
= OCT 000060746134,
COMMAND FORMAT (4) = OCT 103060606023, OCT -064444130274,
FIRST FORMAT •
FIRST PRINT VARIABLE = 32765,
FIXED POINT FORMAT (9) = OCT 260200330773, OCT 260200330173,
OCT 260200330273, OCT 260200330373, OCT 260200330473,
OCT 260200330573, OCT 260200330673, OCT 260200330773,
OCT 260200330073,
FLOAT CONSTANT = OCT 233000000000,
FLOATING POINT FORMAT = OCT 250200330773,
FLOATING POINT TWO = 2.0*0,
FLOATING POINT TEN = 10.0*0,
LAST FORMAT WORD = OCT 34777777777777777,
LAST PRINT CALL WORD = OCT 0021000700111,
LIST ALL COMMANDS = 1,
LOCATION FORMAT (4) = OCT 010530606060, OCT 234644442145,
OCT 246043462313, OCT -050573606060,
LOG E 2 = 0.69314718*0,
MANUAL = 0,
MAXIMUM FIXED POINT NUMBER = 1.0*10,
MINUS = OCT -006060606060,
OCTAL FORMAT (2) = OCT 036704304623, OCT -231342010373,
ONETAG = OCT 100000,
OP CODE MASK = OCT 177,
OUTPUT CONSTANT = OCT 65000,
PERFORMED = 1,
TAB FORMAT CLN LBR TB (6=23) RBR = OCT -206000000067,
TABL FORMAT = OCT 010567450573,

LOG E 2 = 0.69314718*0,
MANUAL = 0,
MAXIMUM FIXED POINT NUMBER = 1.0*10,
MINUS = OCT -006060606060,
OCTAL FORMAT (2) = OCT 036704304623, OCT -231342010373,
ONETAG = OCT 100000,
OP CODE MASK = OCT 177,
OUTPUT CONSTANT = OCT 65000,
PERFORMED = 1,
TAB FORMAT CLN LBR TB (6=23) RBR = OCT -206000000067,
TABL FORMAT = OCT 010567450573,
OPERATING NUMBERS
FORTY TWO = 42, COMMAND SWITCH
FIFTY FIVE = 55,
FLOWCHART NUMBERS
SIXTY FIVE = 65,
SEVENTY ONE = 71,
SEVENTY SEVEN = 77,
SEVENTY EIGHT = 88,
SEVENTY NINE = 89,
TEN THOUSAND = 10000,
ONE HUNDRED = 100,
F变动I ONE HUNDRED = 102,
TEN THOUSAND = 10000,
ONE THOUSAND = 1000,
ONE HUNDRED = 100,
TEN = 10,
ONE = 1,
IF = 1,
FOR = 1,
WHILE = 1,
READ CARD = 1,
READ CLOCK = 1,
READ = 1,
PRINT = 1,
WRITE = 1,
INPUT = 1,
PRINT = 1,
READ = 1,
WRITE = 1,
INPUT = 1,
END = 1,
PROCEDURE = 1,
SUBROUTINE = 1,
FUNCTION = 1,
DIMENSIONING CLN
ACCUMULATOR ADDRESS CLN = MCH 0000000 ACCUMULATOR,
DATUM ADDRESS CLN = MCH 1000000 DATUM,
ENTER ADDRESS CLN = MCH 0002000 ENTER,
FORMAT ADDRESS CLN = MCH 0000200 FORMAT,
LOCATION ADDRESS CLN = MCH 1000000 LOCATION,
TABL ADDRESS CLN = MCH 1000000 TABL,
TRANFER TO EXIT RC CLN = MCH 0020000 EXIT RETURN CARRIAGE,
W DIFFERENCE ADDRESS CLN = MCH 0000000 W DIFFERENCE,
WD ADDRESS CLN = MCH 1000000 WD,
GRAPH ADDRESS CLN = MCH 1000000 GRAPH,
CHL ADDRESS CLN = MCH 0000000 CHL,
OP CODE ADDRESS CLN = MCH 1000000 OP CODE,
INDEX ADDRESS CLN = MCH 1000000 INDEX,
TRANSFRET ON MINUS,
FORMAT CLN = MCH 7460600 OCT 0060060,
GO TO IF = MCH 0000000 0,
GO TO FOR = MCH 0000000 0,
GO TO WHILE = MCH 0000000 0,
GO TO READ CARDS = MCH 0000000 0,
GO TO READ CLOCK = MCH 0000000 0,
GO TO READ = MCH 0000000 0,
GO TO WRITE = MCH 0000000 0,
GO TO INPUT = MCH 0000000 0,
EXECUTE CLN = MCH 0000000 0,
EXIT COMMAND CLN = END
OPERATING MODES AND EXECUTE COMMAND SWITCH

FLOWCHART NUMBER 00003

$PERMIT MANUAL OPERATION CLN (COMMENT OP CODE = 67)
IF TAB NEQ ONE
THEN DO RETURN CARRIAGE, $$
MANUAL = MODE, IOH PRINT (8,0),
READ INSTRUCTION CLN
DO READ CARD, DO COMMAND CARD CONVERSION, IOH PRINT (4,0, INDEX,
OP CODE, ADDR, C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11), PERFORMED
ADDRESS SEPARATION, DO EXECUTE COMMAND, GO TO READ INSTRUCTION.

PROCEDURE READ CARD CLN
BEGIN IOH READ (1,0, C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11, COL 64,
COL 66, COL 68, COL 69, COL 70, COL 72, COL 74, COL 76, SIGN, EXECUTE),
FOR J EQU 0 STEP 1 UNTIL 1
BEGIN IF COL 64 LBK J RBK EQU BLANK
THEN ZERO = COL 64 LBK J RBK $$ END t END ,

COMPUTE AUTOMATICALLY CLN (COMMENT OP CODE = 69)
IF TAB NEQ ONE
THEN DO RETURN CARRIAGE, $$
AUTOMATIC = MODE, IOH PRINT (9,0), DO GET ADDRESS, I = LOCATION,
COMMAND EXECUTION CLN
MEMORY LBK LOCATION RBK AND OP CODE MASK = OP CODE,
DO EXECUTE COMMAND, LOCATION + ONE = LOCATION,
GO TO COMMAND EXECUTION.

PROCEDURE EXECUTE COMMAND CLN
BEGIN GO TO EXECUTE LBK OP CODE RBK .
GO TO BLOCK COPY. GO TO ENDJOB. GO TO COPY IRA INTO IRD.
GO TO CLEAR IRA AN ADD IRD. GO TO INCREMENT C BASE.
GO TO INCREMENT W BASE. GO TO ASSIGN C LIMIT.
GO TO ASSIGN C DIFFERENCE. GO TO ASSIGN C BASE.
GO TO ASSIGN W LIMIT. GOTO ASSIGN W DIFFERENCE. GOTO ASSIGN W BASE.
GO TO COMPUTE AUTOMATICALLY. GO TO BREAKPOINT HALT.
GO TO PERMIT MANUAL OPERATION. GO TO STORE MQ. GO TO LOAD MQ.
GO TO CLOCK. GO TO RING BELL. GO TO STOP SELECTIVE PRINT.
GO TO INITIATE SELECTIVE PRINT. GO TO ERROR. GO TO ERROR.
GO TO ERROR. GO TO ERROR. GO TO ERROR. GO TO READ PAPER CARDS.
GO TO ERROR. GOTO ERROR. GOTO PERMIT TYPE IN OF FLOATING POINT DATA.
GO TO ERROR. GOTO PERMIT COMMAND TYPE IN. GOTO STORE. GOTO DIVIDE.
GO TO INVERSE DIVIDE. GOTO ERROR. GOTO CLEAR AN ADD ABOSOLUTE VALUE.
GO TO MULTIPLY. GO TO ADD. GO TO CLEAR AN ADD.
GO TO SUBTRACT. GO TO CLEAR AN SUBTRACT. GO TO PUNCH PAPER CARDS.
GO TO TYPE FIX POINT NUMBER AN RC. GOTO TYPE MEMORY IN OCTAL AN TAB.
GO TO ERROR. GO TO TYPE COMMAND FROM MEMORY AN TAB.
GO TO TYPE FLOAT POINT NUMBER AN RC.
GO TO TYPE FIXED POINT NUMBER AN TAB.
GO TO TYPE FLOATING POINT NUMBER AN TAB.
GO TO TYPE TABULATING NUMBER AN TAB. GOTO POSITION TYPEWRITER PAPER.
GO TO TRANSFER. GO TO MARK PLACE 2 AN TRANSFER.
GO TO ERROR. GO TO MARK PLACE 1 AN TRANSFER. GO TO ERROR.
GO TO ERROR. GO TO TRANSFER ON ZERO. GO TO TRANSFER ON MINUS.
GO TO ERROR. GO TO TRANSFER ON PLUS AN ZERO.
GO TO ERROR. GO TO RETURN TO MARK PLACE 2.
GO TO ERROR. GO TO RETURN TO MARKED PLACE 1. GO TO ERROR.
GO TO ERROR. GO TO ERROR. GO TO ERROR. GO TO ERROR.
GO TO SET IRA. GO TO PERFORM SUBROUTINE. GO TO LOAD SUBROUTINES.
GO TO TYPE LOCATION OF LAST COMMAND EXECUTED. GO TO ERROR.
GO TO ERROR. GO TO ERROR. GO TO ERROR. GO TO ERROR.

EXECUTE CLN GO TO NO OPERATION.
EXIT COMMAND CLN END ;;
$PERMIT COMMAND TYPE IN CLN
   IF TAB NEQ ONE THEN DO RETURN CARRIAGE, $$
   IOH PRINT (6,0),
READ COMMAND CLN
   DO READ CARD, DO GET ADDRESS, I = ADDRESS,
   DO COMMAND CARD CONVERSION, IF EXECUTE EQU BLANK
   THEN IOH PRINT (3,0, ADDRESS, INDEX, OP CODE, ADDR, C1,C2,
   C3,C4,C5,C6,C7,C8,C9,C10,C11), CHL * 2 EXP 18 + WD
   * 2 EXP 11 + INDEX * 2 EXP 7 + OP CODE = MEMORY LBK | RBK $
   ELSE IOH PRINT (4,0, INDEX, OP CODE, ADDR, C1,C2,C3,C4,C5,
   C6,C7,C8,C9,C10,C11), PERFORMED = ADDRESS SEPARATION,
   DO EXECUTE COMMAND, ADDRESS - ONE = ADDRESS $
CHECK MODE CLN
   IF MODE EQU MANUAL THEN ADDRESS + ONE = ADDRESS, GO TO READ COMMAND.
   ELSE GO TO EXIT COMMAND.
READ COMMANDS CLN
   DO READ CARD, DO COMMAND CARD CONVERSION, IF EXECUTE EQU BLANK
   THEN IOH PRINT (3,0, ADDRESS, INDEX, OP CODE, ADDR, C1,C2,
   C3,C4,C5,C6,C7,C8,C9,C10,C11), CHL * 2 EXP 18 + WD
   * 2 EXP 11 + INDEX * 2 EXP 7 + OP CODE = MEMORY LBK ADDRESS RBK , ADDRESS + ONE = ADDRESS,
   GO TO READ COMMAND.
   ELSE IOH PRINT (4,0, INDEX, OP CODE, ADDR, C1,C2,C3,C4,C5,
   C6,C7,C8,C9,C10,C11), PERFORMED = ADDRESS SEPARATION,
   DO EXECUTE COMMAND, GO TO READ COMMAND.
PERMIT TYPE IN OF FLOATING POINT DATA CLN
   IF TAB NEQ ONE THEN DO RETURN CARRIAGE, $$
   IOH PRINT (7,0),
READ DATUM CLN
   DO READ CARD, DO GET ADDRESS, I = ADDRESS, IF EXECUTE EQU BLANK
   THEN DO DATUM CARD CONVERSION, IOH PRINT (5,0, ADDRESS,
   DATUM, C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11), DATUM
   = MEMORY LBK I RBK $
   ELSE DO COMMAND CARD CONVERSION, IOH PRINT (4,0, INDEX,
   OP CODE, ADDR, C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11),
   PERFORMED = ADDRESS SEPARATION, DO EXECUTE COMMAND,
   ADDRESS - ONE = ADDRESS $
CHK MODE CLN
   IF MODE EQU MANUAL
   THEN ADDRESS + ONE = ADDRESS, GO TO READ DATA.
   ELSE GO TO EXIT COMMAND.
READ DATA CLN
   DO READ CARD, IF EXECUTE EQU BLANK
   THEN DO DATUM CARD CONVERSION, IOH PRINT (5,0, ADDRESS,
   DATUM, C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11), DATUM
   = MEMORY LBK ADDRESS RBK , ADDRESS + ONE = ADDRESS,
   GO TO READ DATA.
   ELSE DO COMMAND CARD CONVERSION, IOH PRINT (4,0, INDEX,
   OP CODE, ADDR, C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11),
   PERFORMED = ADDRESS SEPARATION, DO EXECUTE COMMAND,
   GO TO READ DATA..
$CLEAR
AN SUBTRACT CLN ZERO CLN
DO GET ADDRESS, ZERO - MEMORY LBK I RBK = ACCUMULATOR,
GO TO EXIT COMMAND.

SUBTRACT CLN
DO GET ADDRESS, ACCUMULATOR - MEMORY LBK I RBK = ACCUMULATOR,
GO TO EXIT COMMAND.

CLEAR AN ADD CLN
DO GET ADDRESS, MEMORY LBK I RBK = ACCUMULATOR, GO TO EXIT COMMAND.

ADD CLN
DO GET ADDRESS, ACCUMULATOR + MEMORY LBK I RBK = ACCUMULATOR,
GO TO EXIT COMMAND.

MULTIPLY CLN
DO GET ADDRESS, ACCUMULATOR * MEMORY LBK I RBK = ACCUMULATOR,
GO TO EXIT COMMAND.

INVERSE DIVIDE CLN
DO GET ADDRESS, MEMORY LBK I RBK / ACCUMULATOR = ACCUMULATOR,
GO TO EXIT COMMAND.

DIVIDE CLN
DO GET ADDRESS, ACCUMULATOR / MEMORY LBK I RBK = ACCUMULATOR,
GO TO EXIT COMMAND.

STORE CLN
DO GET ADDRESS, ACCUMULATOR = MEMORY LBK I RBK, GOTO EXIT COMMAND.
TRANSFER OF CONTROL COMMANDS

FLOWCHART NUMBER 00006

$TRANSFER ON PLUS AN ZERO CLN
   (COMMENT OP CODE = 20)
   IF ACCUMULATOR GEQ ZERO
   THEN DO GET ADDRESS, I-ONE = LOCATION, GO TO EXIT COMMAND.
   ELSE GO TO EXIT COMMAND.

TRANSFER ON MINUS CLN
   (COMMENT OP CODE = 22)
   IF ACCUMULATOR LSS ZERO
   THEN DO GET ADDRESS, I-ONE = LOCATION, GO TO EXIT COMMAND.
   ELSE GO TO EXIT COMMAND.

TRANSFER ON ZERO CLN
   (COMMENT OP CODE = 23)
   IF ACCUMULATOR EQU ZERO
   THEN DO GET ADDRESS, I-ONE = LOCATION, GO TO EXIT COMMAND.
   ELSE GO TO EXIT COMMAND.

TRANSFER CLN
   (COMMENT OP CODE = 29)
   DO GET ADDRESS, I-ONE = LOCATION, GO TO EXIT COMMAND.

MARK PLACE 1 AN TRANSFER CLN
   (COMMENT OP CODE = 26)
   LOCATION = MARK PLACE 1, DO GET ADDRESS, I-ONE = LOCATION,
   GO TO EXIT COMMAND.

MARK PLACE 2 AN TRANSFER CLN
   (COMMENT OP CODE = 28)
   LOCATION = MARK PLACE 2, DO GET ADDRESS, I-ONE = LOCATION,
   GO TO EXIT COMMAND.

RETURN TO MARKED PLACE 1 CLN
   (COMMENT OP CODE = 16)
   MARK PLACE 1 = LOCATION, GO TO EXIT COMMAND.

RETURN TO MARKED PLACE 2 CLN
   (COMMENT OP CODE = 18)
   MARK PLACE 2 = LOCATION, GO TO EXIT COMMAND.

CLEAR IRA AN ADD IR O CLN
   (COMMENT OP CODE = 78)
   DO COMMAND SEPARATION, AND = ELEVEN + INDEX = I,
   W DIFFERENCE LBK I RBK = IRA, GO TO EXIT COMMAND.

COPY IRA INTO IR O CLN
   (COMMENT OP CODE = 79)
   DO COMMAND SEPARATION, AND = ELEVEN + INDEX = I,
   IRA = W DIFFERENCE LBK I RBK, GO TO EXIT COMMAND.

SET IRA CLN
   (COMMENT OP CODE = 09)
   DO GET ADDRESS, I = IRA, GO TO EXIT COMMAND.
INDEX REGISTER COMMANDS

FLOWCHART NUMBER 00007

$ASSIGN W BASE CLN LAST COMMAND EXECUTED CLN (COMMENT OP CODE = 70)
DO COMMAND SEPARATION, ADDR = W BASE LBK INDEX RBK,
GO TO EXIT COMMAND.

ASSIGN W DIFFERENCE CLN CLN (COMMENT OP CODE = 71)
DO COMMAND SEPARATION, ADDR = W DIFFERENCE LBK INDEX RBK,
GO TO EXIT COMMAND.

ASSIGN W LIMIT CLN CLN (COMMENT OP CODE = 72)
DO COMMAND SEPARATION, ADDR = W LIMIT LBK INDEX RBK,
GO TO EXIT COMMAND.

INCREMENT W BASE CLN (COMMENT OP CODE = 76)
DO COMMAND SEPARATION, INDEX = I, IF W BASE LBK I RBK
+ W DIFFERENCE LBK I RBK = W BASE LBK I RBK LEQ W LIMIT LBK I RBK,
THEN ADDR - ONE = LOCATION, GO TO EXIT COMMAND.

INCREMENT C BASE CLN (COMMENT OP CODE = 77)
DO COMMAND SEPARATION, INDEX = I, IF C BASE LBK I RBK
+ C DIFFERENCE LBK I RBK = C BASE LBK I RBK LEQ C LIMIT LBK I RBK,
THEN ADDR - ONE = LOCATION, GO TO EXIT COMMAND.

CLEAR IRA AN ADD IR D CLNR AN TAB CLN (COMMENT OP CODE = 78)
DO COMMAND SEPARATE, WD * ELEVEN + INDEX = I,
W DIFFERENCE LBK I RBK = NIRA, GO TO EXIT COMMAND.

COPY IRA INTO IR D CLNR CLN (COMMENT OP CODE = 79)
DO COMMAND SEPARATE, WD * ELEVEN + INDEX = I,
IRA = W DIFFERENCE LBK I RBK, GO TO EXIT COMMAND.

SET IRA CLN POINT NUMBER AN TAB CLN (COMMENT OP CODE = 09)
DO GET ADDRESS, I = IRA, GO TO EXIT COMMAND.
LOAD FP NUM ADDRESS IN PRINTOUT CALL CLN
DATUM ADDRESS - TAB = PRINTOUT LBK L RBK, L = ONE = L.
LOAD FP NUM FORMAT CLN
IF DATUM LBK TAB RBK < LSS MAXIMUM FIXED POINT NUMBER
THEN FIXED POINT FORMAT = FORMAT LBK W RBK 
ELSE FLOATING POINT FORMAT = FORMAT LBK W RBK 
M - ONE = M, TAB + ONE = TAB, GO TO EXIT COMMAND.
$TYPE LOCATION OF LAST COMMAND EXECUTED CLN (COMMENT: OP CODE = 06)
LOCATION - ONE = ADDRESS LBK TAB RBK,' LOAD LOCATION VARIABLE ADDRESS IN PRINTOUT CALL CLN
LOCATION ADDRESS - TAB = PRINTOUT LBK L RBK, L - ONE = L, LOAD LOCATION FORMAT CLN
FOR J EQU 0 STEP 1 UNTIL 3 DO
BEGIN LOCATION FORMAT LBK J RBK = FORMAT LBK M RBK, M - ONE = M, END,
TAB + ONE = TAB, GO TO EXIT COMMAND.

TYPE OUTPUT COMMANDS FOR TYPEWRITER PAPER CLN (COMMENT: OP CODE = 30)
DO GET ADDRESS, I = STORAGE / ONE HUNDRED = CHL * ONE HUNDRED = NUMBER, STORAGE = NUMBER = WD, IF WD NEQ ZERO
THEN BEGIN IF WD GEQ TEN THEN WD / TEN = BCD NUMBER * TEN = NUMBER, BCD NUMBER * 2 EXP 6 + WD - NUMBER = WD $$$ = PRINTOUT LBK LBK RBK, M - ONE = M, DO RETURN CARRIAGE, END $ ELSE TAB + CHL = TAB, GO TO EXIT OP CODE 30.
IF CHL NEQ ZERO THEN CHL + ONE = TAB $
ELSE GO TO EXIT COMMAND.
EXIT OP CODE 30 CLN
CHL * TWENTY = NUMBER, IF NUMBER EQU ONE HUNDRED THEN BCD 100 = TB $ ELSE NUMBER / TEN # 2 EXP 6 = TB $ TAB FORMAT = FORMAT LBK M RBK, M-ONE=M, GO TO EXIT COMMAND.

TYPE TABULATING NUMBER AN TAB CLN RBK (COMMENT: OP CODE = 31)
DO GET ADDRESS, I = TABL LBK TAB RBK, LOAD TABL ADDRESS IN PRINTOUT CALL CLN
TABL ADDRESS - TAB = PRINTOUT LBK L RBK, L - ONE = L, LOAD TABL FORMAT CLN
TABL FORMAT = FORMAT LBK M RBK, M - ONE = M, TAB + ONE = TAB, GO TO EXIT COMMAND.

TYPE FLOATING POINT NUMBER AN TAB CLN (COMMENT: OP CODE = 32)
DO GET ADDRESS, MEMORY LBK I RBK = DATUM LBK TAB RBK, LOAD FL PT NUM ADDRESS IN PRINTOUT CALL CLN
DATUM ADDRESS - TAB = PRINTOUT LBK L RBK, L - ONE = L, LOAD FL PT NUM FORMAT CLN
FLOATING POINT FORMAT = FORMAT LBK M RBK, M - ONE = M, TAB + ONE = TAB, GO TO EXIT COMMAND.

TYPE FIXED POINT NUMBER AN TAB CLN (COMMENT: OP CODE = 33)
DO GET ADDRESS, MEMORY LBK I RBK = DATUM LBK TAB RBK, LOAD FP NUM ADDRESS IN PRINTOUT CALL CLN
DATUM ADDRESS - TAB = PRINTOUT LBK L RBK, L - ONE = L, LOAD FP NUM FORMAT CLN
IF DATUM LBK TAB RBK LSS MAXIMUM FIXED POINT NUMBER THEN FIXED POINT FORMAT = FORMAT LBK M RBK $$ ELSE FLOATING POINT FORMAT = FORMAT LBK M RBK $$ M - ONE = M, TAB + ONE = TAB, GO TO EXIT COMMAND.
FLOWCHART NUMBER 00011

$TYPE FLOAT POINT NUMBER AN RC CLN  (COMMENT : OP CODE = 34)
DO GET ADDRESS, MEMORY LBK I RBK = DATUM LBK TAB RBK,
LOAD FP LNUM ADDRESS IN PRINTOUT CALL CLN
DATUM ADDRESS - TAB = PRINTOUT LBK L RBK, L - ONE = L,
LOAD FP LNUM FORMAT CLN
FLOATING POINT FORMAT = FORMAT LBK M RBK, M - ONE = M,
DO RETURN CARRIAGE, GO TO EXIT COMMAND.

TYPE COMMAND FROM MEMORY AN TAB CLN       (COMMENT : OP CODE = 35)
DO GET ADDRESS, CELL LBK I RBK (7=10) = INDEX LBK TAB RBK,
CELL LBK I RBK AND OP CODE MASK = OP CODE LBK TAB RBK,
CELL LBK I RBK (18=26) = CHL LBK TAB RBK, CELL LBK I RBK (11=17)
= WD LBK TAB RBK,
LOAD COMMAND VARIABLE ADDRESSES IN PRINTOUT CALL CLN
FOR J EQU 0 STEP 1 UNTIL 3 DO
   BEGIN INDEX ADDRESS LBK J RBK - TAB = PRINTOUT LBK L RBK,
   L - ONE = L, END,
LOAD COMMAND FORMAT CLN
FOR J EQU 0 STEP 1 UNTIL 3 DO
   BEGIN COMMAND FORMAT LBK J RBK = FORMAT LBK M RBK,
   M - ONE = M, END,
   TAB + ONE = TAB, GO TO EXIT COMMAND.

TYPE MEMORY IN OCTAL AN TAB CLN             (COMMENT : OP CODE = 37)
DO GET ADDRESS, MEMORY LBK I RBK = DATUM LBK TAB RBK,
LOAD OCTAL VARIABLE ADDRESS IN PRINTOUT CALL CLN
DATUM ADDRESS - TAB = PRINTOUT LBK L RBK, L - ONE = L,
LOAD OCTAL FORMAT CLN
OCTAL FORMAT = FORMAT LBK M RBK,
OCTAL FORMAT LBK 1 RBK = FORMAT LBK M-1 RBK , M - TWO = M,
TAB + ONE = TAB, GO TO EXIT COMMAND.

TYPE FIX POINT NUMBER AN RC CLN             (COMMENT : OP CODE = 38)
DO GET ADDRESS, MEMORY LBK I RBK = DATUM LBK TAB RBK,
LOAD FP LNUM ADDRESS IN PRINTOUT CALL CLN
DATUM ADDRESS - TAB = PRINTOUT LBK L RBK, L - ONE = L,
LOAD FP LNUM FORMAT CLN
IF DATUM LBK TAB RBK LSS MAXIMUM FIXED POINT NUMBER
THEN FIXED POINT FORMAT = FORMAT LBK M RBK $
ELSE FLOATING POINT FORMAT = FORMAT LBK M RBK $
M - ONE = M, DO RETURN CARRIAGE, GO TO EXIT COMMAND.

PROCEDURE RETURN CARRIAGE CLN
BEGIN LAST PRINT CALL WORD = PRINTOUT LBK L RBK ,
LAST FORMAT WORD = FORMAT LBK M RBK , TRANSFER TO EXIT RC
 = PRINTOUT LBK L-1 RBK ,
PRINTOUT CLN
IOH PRINT (10,0, EW,EW,EW,EW,EW,EW,EW,EW,EW,EW,EW,EW,EW,EW,EW,EW,EW,EW,EW,EW,EW,EW,EW,EW,EW),
EXIT RETURN CARRIAGE CLN
ONE = TAB, FIRST PRINT VARIABLE = L, FIRST FORMAT = M, END ,..
$PUNCH PAPER CARDS CLN  (COMMENT OP CODE = 39)
DO GET ADDRESS, I = STORAGE, ONETAG - STORAGE - ADDR INCREMENT
= ORIGIN, STORAGE / ONE HUNDRED * ONE HUNDRED + ADDR INCREMENT
= STORAGE, ONETAG - STORAGE = STORAGE * 2 EXP 18 + ORIGIN = LIMITS,
DO PUNCH CARDS, GO TO EXIT COMMAND.
PROCEDURE PUNCH CARDS CLN
FOR BEGIN DO PUNCH,
LIMITS CLN EMPTY WORD, 
ORIGIN CLN MCH 0000000 OCT 100, END,
GO TO EXIT COMMAND.
READ PAPER CARDS CLN  (COMMENT OP CODE = 55)
DO READ CARDS, EMPTY WORD, GO TO ENDOFFILE RETURN.
GO TO ERROR RETURN. GO TO EXIT COMMAND.
ENDOFFILE RETURN CLN  (COMMENT OP CODE = 00)
IF TAB NEQ ONE, GO TO ENDJB.
THEN DO RETURN CARRIAGE, $$
NO OPERATIONS CLN
IOH PRINT (14,0,), GO TO ENDJB.
ERROR RETURN CLN
IF TAB NEQ ONE
RING BEGIN THEN DO RETURN CARRIAGE, $$
IF TAB IOH PRINT (15,0,), GO TO ENDJB.
THEN DO RETURN CARRIAGE, $$
IOH PRINT (11,0,), GO TO EXIT COMMAND.
CLOCK CLN  (COMMENT OP CODE = 64)
DO READ CLOCK, MCH 000100G ACCUMULATOR, GO TO EXIT COMMAND.
LOAD NO CLN  (COMMENT OP CODE = 65)
DO GET ADDRESS, MEMORY LBK 1 RBK = NO, GO TO EXIT COMMAND.
STORE NO CLN
DO GET ADDRESS, NO = MEMORY LBK 1 RBK, GO TO EXIT COMMAND.
BREAKPOINT HALT CLN
IF TAB NEQ ONE
THEN DO RETURN CARRIAGE, $$
IOH PRINT (12,0,), GO TO ENDJB.
ENDJOB CLN  (COMMENT OP CODE = 80)
IF TAB NEQ ONE
THEN DO RETURN CARRIAGE, $$
ZERO = LIMITS, GO PUNCH CARDS, GO TO ENDJO.
SPECIAL COMMANDS

FLOWCHART NUMBER 00013

$BLOCK COPY CLN ES CLN (COMMENT OP CODE = 81)

DO COMMAND SEPARATION, ADDR - ONE = LAST WORD,
INDEX * ONE HUNDRED + EIGHT HUNDRED = J,
FOR I EQU CHL STEP 1 UNTIL LAST WORD DO
BEGIN MEMORY LBK I RBK = MEMORY LBK J RBK , J + ONE = J, END ,
CHL - EIGHT HUNDRED / ONE HUNDRED = I, INDEX = J,
FOR K EQU 0 STEP 1 UNTIL 5 DO
BEGIN W DIFFERENCE LBK I RBK = W DIFFERENCE LBK J RBK ,
I + ELEVEN = I, J + ELEVEN = J, END ,
GO TO EXIT COMMAND.
ERROR CLN TO CALL SIN COS. GO TO CALL POWER, GO TO CALL LOG.
IF TAB NEQ ONE THEN DO RETURN CARRIAGE, $$
THEN DO RETURN CARRIAGE, $$
ERROR CLN (COMMENT OP CODE = 63)

NO OPERATION CLN (COMMENT OP CODE = 00)
GO TO EXIT COMMAND.
RING BELL CLN (COMMENT OP CODE = 63)
IF TAB NEQ ONE THEN DO RETURN CARRIAGE, $$

CLOCK CLN (COMMENT OP CODE = 64)
DO READ CLOCK, MCH 0601000 ACCUMULATOR, GO TO EXIT COMMAND.

LOAD MQ CLN (COMMENT OP CODE = 65)
DO GET ADDRESS, MEMORY LBK I RBK = MQ, GO TO EXIT COMMAND.

STORE MQ CLN (COMMENT OP CODE = 66)
DO GET ADDRESS, MQ = MEMORY LBK I RBK, GO TO EXIT COMMAND.

BREAKPOINT HALT CLN (COMMENT OP CODE = 68)
IF TAB NEQ ONE THEN DO RETURN CARRIAGE, $$
CALL IOH PRINT (12,0), GO TO ENDJB. (COMMENT N = 2)

ENDJOB CLN (COMMENT OP CODE = 80)
IF TAB NEQ THEN DO RETURN CARRIAGE, $$
CALL IOH PRINT (22,0), CHL, GO TO READ SUBROUTINE.
CALL ARCTAN CLN (COMMENT N = 47)
CALL HYPERBOLICS CLN (COMMENT N = 47)
CALL INGEN REGISTER UTILIZATION CLN (COMMENT N = 12)
CALL ERRORS CLN (COMMENT N = 12)

IOH PRINT (264, OP CODE), GO TO ENDJB.
LOADING SUBROUTINES

$LOAD SUBROUTINES CLN

(comment: OP CODE = 07)

IF TAB NEQ ONE
THEN DO RETURN CARRIAGE, $$

IOH PRINT (16,0,)

READ SUBROUTINE CLN

DO READ CARD, DO COMMAND CARD CONVERSION,
GO TO CALL SUBROUTINE LBK OP CODE RBK.

IF SUBROUTINE EQQ 190 ACUMULATOR,

GO TO CALL INDEX REGISTER UTILIZATION.

IF SUBROUTINE EQQ 190 ACUMULATOR,

GO TO CALL ERROR. GO TO CALL ERROR.

GO TO CALL ERROR. GO TO CALL HYPERBOLICS. GO TO CALL ARCTAN.

GO TO CALL SIN COS. GO TO CALL POWER. GO TO CALL LOG.

CALL SUBROUTINE CLN GO TO EXIT LOADING SUBROUTINES.

EXIT LOADING SUBROUTINES CLN

COMMENT OP CODE = 07

IF SIGN EQU MINUS
THEN FIXED POINT FORMAT LBK WD RBK = FIXED POINT FORMAT,

IOH PRINT (17,0, WD), GO TO EXIT COMMAND.$

IF WD EQU ZERO
THEN IOH PRINT (18,0), GO TO EXIT COMMAND.$

IF WD EQU ONE
THEN IOH PRINT (29,0), GO TO READ SUBROUTINE.$

IF WD EQU TWO
THEN IOH PRINT (34,0, FOR J EQU 0 STEP 1 UNTIL 65 DO
BEGIN ZERO = W DIFFERENCE LBK J RBK , END ,

GO TO READ SUBROUTINE.$

IF WD EQU THREE
THEN IOH PRINT (35,0, FOR J EQU 0 STEP 1 UNTIL 65 DO
BEGIN ZERO = MEMORY LBK J RBK , END ,

GO TO READ SUBROUTINE.$

CALL FRACTION SELECTOR CLN

IOH PRINT (28,0, CHL), CHL = FRAC SEL CHANNEL,

GO TO READ SUBROUTINE.

CALL SQUARE ROOT CLN

(COMMENT N = 2)

IOH PRINT (19,0, CHL), GO TO READ SUBROUTINE. 0601000 IRA.

CALL LOG CLN

(COMMENT N = 3)

IOH PRINT (20,0, CHL), CHL = LOG CHANNEL, GO TO READ SUBROUTINE.

CALL POWER CLN

(COMMENT N = 4)

IOH PRINT (21,0,CHL), CHL = POWER CHANNEL, GO TO READ SUBROUTINE.

CALL SIN COS CLN

(COMMENT N = 5)

IOH PRINT (22,0, CHL), GO TO READ SUBROUTINE.

CALL ARCTAN CLN

(COMMENT N = 6)

IOH PRINT (23,0, CHL), GO TO READ SUBROUTINE.

CALL HYPERBOLICS CLN

(COMMENT N = 7)

IOH PRINT (27,0, CHL), GO TO READ SUBROUTINE.

CALL INDEX REGISTER UTILIZATION CLN

(COMMENT N =12)

IOH PRINT (25,0, CHL), CHL = IRU CHANNEL, GO TO READ SUBROUTINE.

CALL ERROR CLN

IOH PRINT (26,0, OP CODE), GO TO ENDJB.

IF CHL EQU POWER CHANNEL.

THEN MCH 050000 FLOATING POINT TNO, MCH 0560000 ACUMULATOR,

GO POWER 2, MCH 0601000 ACUMULATOR, GO TO EXIT COMMAND.$

IF CHL EQU FRAG SEL CHANNEL.

THEN GO TO SELECT FRACTION.
PERFORMING SUBROUTINES

FLOWCHART NUMBER 00015

$PERFORM SUBROUTINE CLN (COMMENT OP CODE = 08)

DO GET ADDRESS, I = STORAGE / ONE HUNDRED = CHL * ONE HUNDRED

= NUMBER, STORAGE-NUMBER = SUBROUTINE, IF SUBROUTINE LEQ TWENTY SIX

THEN GO TO SUBROUTINE TRANSFER LBK SUBROUTINE RBK.$

IF SUBROUTINE EQU THIRTY NINE (COMMENT SUBROUTINE = WD = 39)

THEN MCH 0500000 ACCUMULATOR, DO SIND, MCH 0601000 ACCUMULATOR,

GO TO EXIT COMMAND.$

IF SUBROUTINE EQU FORTY TWO (COMMENT SUBROUTINE = WD = 42)

THEN MCH 0500000 ACCUMULATOR, DO SIN, MCH 0601000 ACCUMULATOR,

GO TO EXIT COMMAND.$

IF SUBROUTINE EQU SEVENTY ONE (COMMENT SUBROUTINE = WD = 71)

THEN MCH 0500000 ACCUMULATOR, DO LOG 10, MCH 0601000 ACCUMULATOR,

GO TO EXIT COMMAND.$

IF SUBROUTINE EQU SEVENTY TWO (COMMENT SUBROUTINE = WD = 72)

THEN MCH 0500000 FLOATING POINT TEN, MCH 0560000 ACCUMULATOR,

DO POWER 10, MCH 0601000 ACCUMULATOR, GO TO EXIT COMMAND.$

ELSE GO TO ERROR TRANSFER.

GOTO COS TRANSFER. GOTO ATAND TRANSFER.

GOTO ARCTAN TRANSFER. GOTO ERROR TRANSFER. GOTO EXPT TRANSFER.

GOTO ERROR TRANSFER. GOTO ERROR TRANSFER. GOTO ERROR TRANSFER.

GOTO ERROR TRANSFER. GOTO LOG E TRANSFER. GOTO ERROR TRANSFER.

GOTO ERROR TRANSFER. GOTO CUBERT TRANSFER. GOTO EXP3 TRANSFER.

GOTO EXP2 TRANSFER. GOTO EXP1 TRANSFER. GOTO TANH TRANSFER.

GOTO SIND TRANSFER. GOTO CHECK SUBROUTINE. GOTO SELECT FRACTION.

GOTO SELECT FRACTION. GOTO SELECT FRACTION. GOTO SELECT FRACTION.

GOTO SELECT FRACTION. GOTO SELECT FRACTION. GOTO SELECT FRACTION.

GOTO SELECT FRACTION. GOTO SELECT FRACTION.

SUBROUTINE TRANSFER CLN GOTO FIX TRANSFER.

SELECT FRACTION CLN

FIXED POINT FORMAT LBK SUBROUTINE RBK = FIXED POINT FORMAT,

GO TO EXIT COMMAND.

FIX TRANSFER CLN (COMMENT SUBROUTINE = WD = 00)

IF CHL EQU IRU CHANNEL

THEN MCH 0500000 ACCUMULATOR, DO FIX, MCH 0601000 IRA,

GO TO EXIT COMMAND.$

IF CHL EQU FRAC SEL CHANNEL

THEN GO TO SELECT FRACTION.

ELSE GO TO ERROR TRANSFER.

FLOAT TRANSFER CLN (COMMENT SUBROUTINE = WD = 01)

IF CHL EQU IRU CHANNEL

THEN MCH 0500000 IRA, DO FLOAT, MCH 0601000 ACCUMULATOR,

GO TO EXIT COMMAND.$

IF CHL EQU FRAC SEL CHANNEL

THEN GO TO SELECT FRACTION.

ELSE GO TO ERROR TRANSFER.

CHECK SUBROUTINE CLN (COMMENT SUBROUTINE = WD = 08)

IF CHL EQU LOG CHANNEL

THEN MCH 0500000 ACCUMULATOR, DO LOG 2, MCH 0241000 LOG E 2,

MCH 4600000 ACCUMULATOR, GO TO EXIT COMMAND.$

IF CHL EQU POWER CHANNEL

THEN MCH 0500000 FLOATING POINT TWO, MCH 0560000 ACCUMULATOR,

DO POWER 2, MCH 0601000 ACCUMULATOR, GO TO EXIT COMMAND.$

IF CHL EQU FRAC SEL CHANNEL

THEN GO TO SELECT FRACTION.
ELSE GO TO ERROR TRANSFER.
SINH TRANSFER CLN                  (COMMENT SUBROUTINE = WD = 09)
       MCH 050000 ACCUMULATOR, DO SINH COSH, GO TO ERROR TRANSFER.
       MCH 0601000 ACCUMULATOR, MCH 460000 MQ, GO TO EXIT COMMAND.
TANH TRANSFER CLN                  (COMMENT SUBROUTINE = WD = 10)
       MCH 050000 ACCUMULATOR, DO TANH, MCH 0601000 ACCUMULATOR,
       GO TO EXIT COMMAND.
EXP1 TRANSFER CLN                   (COMMENT SUBROUTINE = WD = 11)
       MCH 050000 ACCUMULATOR, MCH 056000 MQ, DO EXP1,
       MCH 0601000 ACCUMULATOR, GO TO EXIT COMMAND.
EXP2 TRANSFER CLN                   (COMMENT SUBROUTINE = WD = 12)
       MCH 050000 ACCUMULATOR, MCH 056000 MQ, DO EXP2,
       UNTIL 1 DO A = 2 EXP 6.
EXP3 TRANSFER CLN                   (COMMENT SUBROUTINE = WD = 13)
       MCH 050000 ACCUMULATOR, MCH 056000 MQ, DO EXP3,
       MCH 0601000 ACCUMULATOR, GO TO EXIT COMMAND.
LOG TRANSFER CLN                    (COMMENT SUBROUTINE = WD = 14)
       MCH 050000 ACCUMULATOR, DO LOG E, MCH 0601000 ACCUMULATOR,
       GO TO EXIT COMMAND.
EXPT TRANSFER CLN                   (COMMENT SUBROUTINE = WD = 22)
       MCH 050000 ACCUMULATOR, DO EXPT, MCH 0601000 ACCUMULATOR,
       GO TO EXIT COMMAND.
COSD TRANSFER CLN                   (COMMENT SUBROUTINE = WD = 23)
       MCH 050000 ACCUMULATOR, DO COSD, MCH 0601000 ACCUMULATOR,
       GO TO EXIT COMMAND.
ARCTAN TRANSFER CLN                 (COMMENT SUBROUTINE = WD = 24)
       MCH 050000 ACCUMULATOR, DO ARCTAN, MCH 0601000 ACCUMULATOR,
       GO TO EXIT COMMAND.
ATAND TRANSFER CLN                  (COMMENT SUBROUTINE = WD = 25)
       MCH 050000 ACCUMULATOR, DO ATAND, MCH 0601000 ACCUMULATOR,
       GO TO EXIT COMMAND.
CUBERT TRANSFER CLN                 (COMMENT SUBROUTINE = WD = 26)
       MCH 050000 ACCUMULATOR, DO CUBE ROOT, MCH 0601000 ACCUMULATOR,
       GO TO EXIT COMMAND.
ERROR TRANSFER CLN                  (COMMENT SUBROUTINE = WD = 26)
       IOH PRINT (2650, SUBROUTINE), GO TO ENDBB.
       THEN GO TO SELECTIVE PRINT TRANSFER.
       ACCUMULATOR = PAST ACCUMULATOR, DO EXECUTE COMMAND,
       DO GET ADDRESS, DO CONVERT COMMAND TO BCD,
       IF SECOND SELECTOR A AND COMMAND A NEQ FIRST SELECTOR A
       THEN GO TO EXIT SELECT COMMANDS.
       IF SECOND SELECTOR B AND COMMAND B NEQ FIRST SELECTOR B
       THEN GO TO EXIT SELECT COMMANDS.
       PRINT SELECTED COMMAND CLN
       I = ADDR, IF ACCUMULATOR NEQ PAST ACCUMULATOR
       THEN IOH PRINT (3150, LOCATION, INDEX, OP CODE, ADDR,
       ACCUMULATOR)
       ELSE IOH PRINT (3250, LOCATION, INDEX, OP CODE, ADDR)
       EXIT SELECT COMMANDS CLN
       LOCATION + ONE = LOCATION, GO TO LIST SELECTED COMMANDS.
       SELECTIVE PRINT TRANSFER CLN
       DO GET ADDRESS, DO CONVERT COMMAND TO BCD,
       IF SECOND SELECTOR A AND COMMAND A NEQ FIRST SELECTOR A
       THEN GO TO EXECUTE TRANSFER COMMAND.
       IF SECOND SELECTOR B AND COMMAND B NEQ FIRST SELECTOR B
       THEN GO TO EXECUTE TRANSFER COMMAND.
       I = ADDR, IOH PRINT (3250, LOCATION, INDEX, OP CODE, ADDR)
       EXECUTE TRANSFER COMMAND CLN
       DO EXECUTE COMMAND, LOCATION + ONE = LOCATION.
SELECTIVE PRINT = GO TO LIST SELECTED COMMANDS.

INITIATE SELECTIVE PRINT CLN $ Step 1 Until 3 (COMMENT: OP CODE = 61)

COMMAND EXECUTION = PROGRAM STORAGE, ENTER SELECTIVE PRINT = COMMAND EXECUTION.

READ CARD SELECTORS CLN

DO READ CARD, IOH PRINT (24,0, COL 64, COL 66, COL 68, COL 69, COL 70, COL 72, COL 74, 61, C3, C4, C5, C6, C7, C8, C9, C10, C11),

CONVERT SELECTOR CLN

COLUMN K = SECOND SELECTOR A, FOR J EQU 0 Step 1 Until 1 DO
BEGIN COLUMN 0 LBK J RBK + SECOND SELECTOR A = 2 EXP 6
END,
COLUMN C = SECOND SELECTOR B, FOR J EQU 0 Step 1 Until 3 DO
BEGIN COLUMN H LBK J RBK + SECOND SELECTOR B = 2 EXP 6
END,

IF SELECTOR EQU ONE THEN TWO = SELECTOR, SECOND SELECTOR A = FIRST SELECTOR A,
SECOND SELECTOR B = FIRST SELECTOR B,
GO TO READ CARD SELECTORS.$

IF SECOND SELECTOR A NEQ ZERO THEN GO TO EXIT COMMAND.$

IF SECOND SELECTOR B EQU ZERO THEN LIST ALL COMMANDS = SECOND SELECTOR $$
GO TO EXIT COMMAND.

START SELECTIVE PRINT CLN

IOH PRINT (30,0,), IF SECOND SELECTOR EQU LIST ALL COMMANDS THEN GO TO LIST PROGRAM.$

LIST SELECTED COMMANDS CLN

IF MEMORY LBK LOCATION RBK AND OP CODE MASK = OP CODE / TEN = BCD NUMBER EQU TWO THEN GO TO SELECTIVE PRINT TRANSFER.$

IF BCD NUMBER EQU ONE THEN GO TO SELECTIVE PRINT TRANSFER.$

IF OP CODE EQU SEVENTY SIX THEN GO TO SELECTIVE PRINT TRANSFER.$

IF OP CODE EQU SEVENTY SEVEN THEN GO TO SELECTIVE PRINT TRANSFER.$

ACCUMULATOR = PAST ACCUMULATOR, DO EXECUTE COMMAND, DO GET ADDRESS, DO CONVERT COMMAND TO BCD,
IF SECOND SELECTOR A AND COMMAND A NEQ FIRST SELECTOR A THEN GO TO EXIT SELECT COMMANDS.$

IF SECOND SELECTOR B AND COMMAND B NEQ FIRST SELECTOR B THEN GO TO EXIT SELECT COMMANDS.$

PRINT SELECTED COMMAND CLN

I = ADDR, IF ACCUMULATOR NEQ PAST ACCUMULATOR THEN IOH PRINT (31,0, LOCATION, INDEX, OP CODE, ADDR, ACCUMULATOR)$
ELSE IOH PRINT (32,0, LOCATION, INDEX, OP CODE, ADDR)$

EXIT SELECT COMMANDS CLN

LOCATION + ONE = LOCATION, GO TO LIST SELECTED COMMANDS.

SELECTIVE PRINT TRANSFER CLN

DO GET ADDRESS, DO CONVERT COMMAND TO BCD,
IF SECOND SELECTOR A AND COMMAND A NEQ FIRST SELECTOR A THEN GO TO EXECUTE TRANSFER COMMAND.$

IF SECOND SELECTOR B AND COMMAND B NEQ FIRST SELECTOR B THEN GO TO EXECUTE TRANSFER COMMAND.$

I = ADDR, IOH PRINT (32,0, LOCATION, INDEX, OP CODE, ADDR), EXECUTE TRANSFER COMMAND CLN

DO EXECUTE COMMAND, LOCATION + ONE = LOCATION,
PROGRAM SUBROUTINE GO TO LIST SELECTED COMMANDS.

PROCEDURE CONVERT COMMAND TO BCD CLN

FLOWCHART BEGIN BCD NUMBER * TEN = NUMBER, INDEX * 2 EXP 12 + BCD NUMBER
* 2 EXP 6 + OPCODE - NUMBER = COMMAND A,
$ PROCEDURE = STORAGE, FOR J EQU 0 STEP 1 UNTIL 3 DO
BEGIN STORAGE / TEN THOUSAND LBK J RBK = BCD NUMBER LBK J RBK
COLUMN 7* TEN THOUSAND LBK J RBK = NUMBER, INDEX
THEN STORAGE - NUMBER = STORAGE, END $ $
BCD NUMBER = COMMAND B, FOR J EQU 1 STEP 1 UNTIL 3 DO ADDR, END
BEGIN COMMAND B * 2 EXP 6 + BCD NUMBER LBK J RBK
PROCEDURE = COMMAND B, END $ $ 
BEGIN COMMAND B * 2 EXP 6 + STORAGE = COMMAND B, END $ $ 
LIST PROGRAM CLN TO LBK J RBK + NUMBER * TEN = NUMBER, END $ $ 
FLO IF MEMORY LBK LOCATION RBK AND OP CODE MASK = OP CODE / TEN
= STORAGE EQU TWO CONSTANT = DATUM, FLOAT CONSTANT = DATUM = DATUM,
THEN GO TO PRINT TRANSFER.$
IF STORAGE EQU ONE = INDEX, THEN GO TO PRINT TRANSFER.$ EXponent DO
IF OP CODE EQU SEVENTY SIX = INDEX, THEN GO TO PRINT TRANSFER.$
IF OP CODE EQU SEVENTY SEVEN = INDEX
THEN GO TO PRINT TRANSFER.$ EXPONENT DO
ACCUMULATOR = PAST ACCUMULATOR, DO EXECUTE COMMAND
THEN IOH PRINT (31,0, LOCATION, INDEX, OP CODE, ADDR, ACCUMULATOR)$
ELSE IOH PRINT (32,0, LOCATION, INDEX, OP CODE, ADDR)$
LOCATION + ONE = LOCATION, GO TO LIST PROGRAM.
BEGIN PRINT TRANSFER CLN
THEN GET ADDRESS, I = ADDR, IF ACCUMULATOR NEQ PAST ACCUMULATOR
THEN IOH PRINT (32,0, LOCATION, INDEX, OP CODE, ADDR, ACCUMULATOR)$
THEN IOH PRINT (32,0, LOCATION, INDEX, OP CODE, ADDR)$
LOCATION + ONE = LOCATION, GO TO LIST PROGRAM.
BEGIN SELECTIVE PRINT CLN (COMMENT OP CODE = 62)
IOH PRINT (33,0,), PROGRAM STORAGE = COMMAND EXECUTION, ZERO
= FIRST SELECTOR A = FIRST SELECTOR B = SECOND SELECTOR A
= SECOND SELECTOR B = SECOND SELECTOR, ONE = SELECTOR,
GO TO EXIT COMMAND.
ELSE, OUTPUT CONSTANT = IN COMP ADDRESS
= N I FERENCE ADDRESS = I, GO TO EXIT GET ADDRESS.$
IF INDEX EQU ZERO
THEN ONE HUNDRED = CHL + WD = I $ $ 
ELSE ONE HUNDRED + CHL + WD + N BASE LBK INDEX RBK
= C BASE LBK INDEX RBK = I $ $ 
EXIT GET ADDRESS CLN
= ADDRESS SEPARATION, END 

PROCEDURE COMMAND SEPARATION CLN
BEGIN IF ADDRESS SEPARATION NEG PERFORMED
THEN CELL LBK LOCATION RBK (7-10) = INDEX
CELL LBK LOCATION RBK (11-26) = ONE HUNDRED = CHL,
CELL LBK LOCATION RBK (11-17) = CHL = ADDR $
ELSE CHL = ONE HUNDRED + CHL + WD = ADDR $ $ END $

PROCEDURE COMMAND SEPARATE CLN
BEGIN IF ADDRESS SEPARATION NEG PERFORMED
THEN CELL LBK LOCATION RBK (17-10) = INDEX
CELL LBK LOCATION RBK (11-26) = CHL,
CELL LBK LOCATION RBK (11-17) = WD $ END$. ...
$ \text{PROCEDURE COMMAND CARD CONVERSION CLN} \\
\text{BEGIN COLUMN 64 = INDEX, COLUMN 66 * TEN + COLUMN 68 = OP CODE, } \\
\text{COLUMN 70 * TEN + COLUMN 72 = CHL, IF COL 69 NEQ BLANK } \\
\text{THEN COLUMN 69 * ONE HUNDRED + CHL = CHL \$} \\
\text{COLUMN 74* TEN + COLUMN 76 = WD, CHL*ONE HUNDRED +WD = ADDR, END \text{,} \text{END\text{,}}$

PROCEDURE \text{DATUM CARD CONVERSION CLN} \\
\text{BEGIN COLUMN 64 * TEN + COLUMN 66 - FIFTY FIVE = EXPONENT, } \\
\text{COLUMN 68 = NUMBER, FOR J EQU 0 STEP 1 UNTIL 3 DO } \\
\text{BEGIN DATUM \text{, FLOATING POINT TEN = DATUM, END ,}$

FLOAT NUMBER CLN
\text{COLUMN 64+ TEN + COLUMN 66 = NUMBER, FOR J EQU 0 STEP 1 UNTIL 3 DO } \\
\text{BEGIN LBK J RBK + NUMBER * TEN = NUMBER, END ,}$

PROCEDURE \text{GET ADDRESS CLN} \\
\text{BEGIN IF ADDRESS SEPARATION NEQ PERFORMED } \\
\text{THEN CELL LBK LOCATION RBK (7=10) = INDEX, } \\
\text{CELL LBK LOCATION RBK (18=26) = CHL, } \\
\text{CELL LBK LOCATION RBK (11=17) = WD \$}$

IF CHL EQU TWENTY ONE \\
\text{THEN OUTPUT CONSTANT - ACCUMULATOR ADDRESS = I, } \\
\text{GO TO EXIT GET ADDRESS.}$

IF WD GEQ ONE HUNDRED TWO \\
\text{THEN WD - ONE HUNDRED TWO = WD * ELEVEN + CHL - EIGHT } \\
\text{WR C IR COMP ADDRESS, OUTPUT CONSTANT + IR COMP ADDRESS } \\
\text{- W DIFFERENCE ADDRESS = I, GO TO EXIT GET ADDRESS.}$

IF INDEX EQ ZERO \\
\text{THEN ONE HUNDRED * CHL + WD = I \$} \text{END\text{,}}$

ELSE ONE HUNDRED * CHL + WD + W BASE LBK INDEX RBK \\
\text{+ C BASE LBK INDEX RBK = I \$}

EXIT GET ADDRESS CLN
\text{ZERO = ADDRESS SEPARATION, END ,}$

PROCEDURE \text{COMMAND SEPARATION CLN} \\
\text{BEGIN IF ADDRESS SEPARATION NEQ PERFORMED } \\
\text{THEN CELL LBK LOCATION RBK (7=10) = INDEX, } \\
\text{CELL LBK LOCATION RBK (18=26) * ONE HUNDRED = CHL, } \\
\text{CELL LBK LOCATION RBK (11=17) + CHL = ADDR \$}$

ELSE CHL * ONE HUNDRED = CHL + WD = ADDR \text{ END ,}$

PROCEDURE \text{COMMAND SEPARATE CLN} \\
\text{BEGIN IF ADDRESS SEPARATION NEQ PERFORMED } \\
\text{THEN CELL LBK LOCATION RBK (7=10) = INDEX, } \\
\text{CELL LBK LOCATION RBK (18=26) = CHL, } \\
\text{CELL LBK LOCATION RBK (11=17) = WD \$ END ,}$
FLOWCHART NUMBER 00020

CONTROL
1(10C6,C3,2(1C1,1X)3C1,5(1X1C1))
2(2XN5,3X2(N2,1X)N5,5X10C6,C3)
3(4H EXECUTE2X2(N2,1X)N5,5X10C6,C3)
4(2XN5,2XE15.7,2X10C6,C3)
5(2XN5,3X2(N2,1X)N5,5X10C6,C3)
6(12H STORE4X10HK OP ADDR)
7(/18H STORE DATA)
8(12H MANUAL MODE///11X10HK OP ADDR)
9(15H AUTOMATIC MODE///)
10(24C1)
11(/11H DING DONG/X)
12(/53H BREAKPOINT HALT NOT ALLOWED. TRANSFERRED TO ENDJOB.)
13(/12H OP CODE = N2,36H NOT DEFINED. TRANSFERRED TO ENDJOB.)
14(/57H END OF FILE WHILE READING CARDS. TRANSFERRED TO ENDJOB.)
15(/51H ERROR WHILE READING CARDS. TRANSFERRED TO ENDJOB.)
16(/18H LOAD SUBROUTINES)
17(35H FIXED POINT FRACTION LENGTH = N1,26H EXIT LOADING SUBROU
1 UINES/X)
18(36H FIXED POINT FRACTION LENGTH = 7 /26H EXIT LOADING SUBROU
1 UINES/X)
19(42H SQUARE ROOT AND CUBE ROOT IN CHANNEL N3)
20(20H LOG IN CHANNEL N3)
21(22H POWER IN CHANNEL N3)
22(24H SIN COS IN CHANNEL N3)
23(23H ARCTAN IN CHANNEL N3)
24(10H SELECTOR 2(1X1C1)1C1,1X5C1,5X10C6,C3)
25(43H INDEX REGISTER UTILIZATION IN CHANNEL N3)
26(17H SUBROUTINE N = N2,36H NOT DEFINED. TRANSFERRED TO ENDJOB.)
27(37H HYPERBOLIC FUNCTIONS IN CHANNEL N3)
28(34H FRACTION SELECTOR IN CHANNEL N3)
29(35H SELECTIVE PRINT IN CHANNEL 8)
30(22H BEGIN SELECTIVE PRINT//11H LOCATION7X7HCOMMAND10X11HACCUMUL
1 ATOR/17X10HK OP ADDR)
31(4XN5,7X2(N2,1X)N5,5XE15.7)
32(4XN5,7X2(N2,1X)N5)
33(/20H END SELECTIVE PRINT/X)
34(17H CLEAR MEMORY)
35(37H CLEAR MEMORY AND INDEX REGISTERS)
Appendix G

A SYNTACTICAL FLOWCHART
for BC NELIAC

As an aid in understanding the syntactical rules of BC NELIAC a flowchart similar to the ALGOL 60 Flowchart has been developed. The shapes of enclosure on the chart have the following meanings:

- **Metalinguistic variables appear in ellipses and indicate the enclosed variable is defined at that place on the chart.**

- **Metalinguistic variables appearing in rectangles means the variable is defined elsewhere on the chart. Grid co-ordinates for the definition appear at the left of the rectangle.**

- **Basic symbols are enclosed in circles.**

- **Vertical arrows indicate a definition of a metalinguistic variable follows.**

- **Horizontal arrows connect the basic symbols metalinguistic variables which form a definition.**

Every metalinguistic formula used to describe BC NELIAC appears on the syntactical flowchart.
BIBLIOGRAPHY

"BC SAP 704 Symbolic Assembly Program," Berkeley, Computer Center, University of California

Feigenbaum, Edward, "Recent Experiments with the EPAM Stimulation of Verbal Learning," Simulations of Cognitive Processes, University of California, 1962


"An Introduction to ALGOL 60 for the B5000 Information Processing System," Detroit, Burroughs Corporation, 1961


Rowe, Alan, "Application of Computer Simulation for Production System Design," Santa Monica, California, Systems Development Corp., 1959

