XI

MONROBOT XI

QUIKOMP PROGRAM SYSTEM

★ ★ **PRELIMINARY EDITION** ★ ★

PROPERTY OF
BOARD OF EDUCATION
CITY OF NEW YORK
Lafayette High School, B'klyn

MONROE

COPYRIGHT © 1965 BY MONROE INTERNATIONAL, INC., A DIVISION OF LITTON INDUSTRIES, ORANGE, N. J. • ALL RIGHTS RESERVED

PRINTED IN U.S.A.

MO-367 ED 8 65

TABLE OF CONTENTS

	PAGE
Elements of the Computer	1
Introduction to the Quikomp Programming System	5
Quikomp Program Functions	10
Numeric Input	11
Numeric Output	14
Data Transfer	21
Program Control	25
Arithmetic Functions	36
Mathematical Functions	39
Quikomp II Program Functions	42
Numeric System	42
Additional Mathematical Functions	43
Operation of the Quikomp Computer	48
Program Input from Typewriter	49
Program Correction	50
Program Operation	50
Preparation of a Data Tape	51
Program Output to Tape	51
Program Input from Add-Punch Tape	52
Operating Procedures	54
Error Control	55
Advanced Programming Information	59
Planning a Program	59
Data Variables and Constants	62
Use of Storage Registers	64

PAGE

Appendices

I Basic Computer Operation 65

II Device Operation 71

1. Input/output Typewriter

2. Paper Tape Reader and Punch

3. Device Connections

III Preparation for Program Input 72

IV Example Problems 74

Insert

Quikomp Functions Card . . . . . Back Cover

ERRATA SHEET

Page 39 The last paragraph should read:

The natural log function finds the log to the base e of the contents of the working register. The logarithm is contained in the working register, the previous contents of the working register are lost. The logarithm of zero or a negative number is illegal.

Page 50 The last line should read:

3. Program operation will begin. See "Operating Procedures" following.

Page 51 Switch 4, item 3 should read:

Type the first numeric data value following the "Operating Procedures" for Quikomp I or II numeric input. The computer will print the data for verification. If the data is correct go to step 4. If the data is incorrect go to step 5.

Page 54 Examples of correct input at <u>bottom</u> of page should read:

1.5 (/) or (T), then (T) or (C/R)

Page 61 At bottom of page, add:

Note: This is Problem 1 in Appendix IV.

ELEMENTS OF THE COMPUTER

The Monrobot XI is a digital computer: it deals with numbers and it can count. The computer can perform simple mathematical operations, make complex calculations, and make logical decisions at a very rapid speed.

The Monrobot XI computing system contains the following elements:

INPUT. The computer accepts information in the form of numbers, that is, decimal digits. The input information may come from a typewriter or a punched paper tape reader.

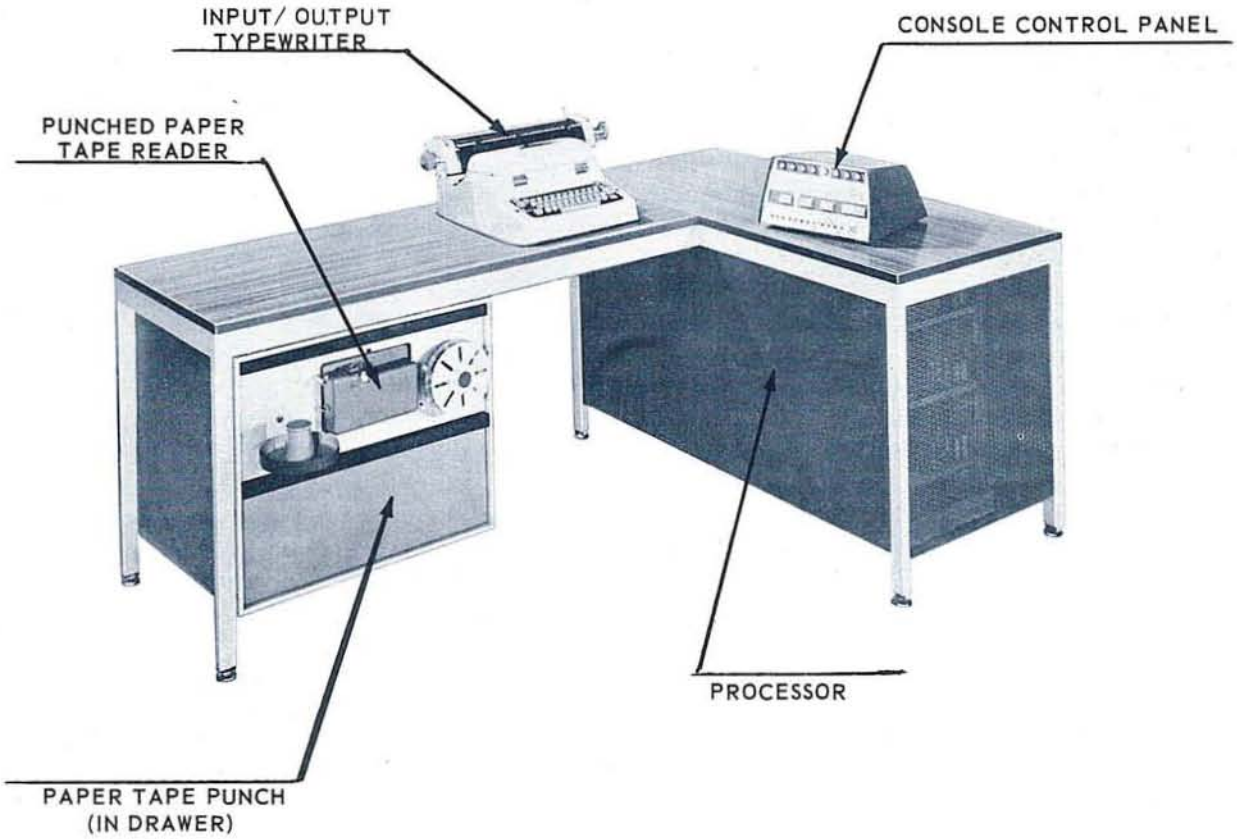
STORAGE. The Monrobot XI has 1,024 or 2,048 storage registers where information or computer instructions may be stored. The storage element is a magnetically recorded, rotating drum which operates on the same principle as a tape recorder. Stored information can be referred to once or many times or be replaced by new information. The information may be original input data, partial results of computations, or tables of computed or reference data.

The storage registers of the computer must also contain a sequence of instructions called a program. These instructions direct the handling of input information, the computations to be performed on the information and the output of results. The concept of stored program instructions is discussed in a subsequent paragraph.

CONTROL. The computer operates under the direction of a sequence of program instructions. This sequence of instructions, each performed in its turn, controls the operation of the computer to cause input of information, computation, and output of results.

ARITHMETIC. The arithmetic unit of the computer provides for addition, subtraction, multiplication, division and comparison of numbers. Complex mathematical functions such as sine, logarithm, or square root are accomplished by combination of the basic arithmetic instructions.

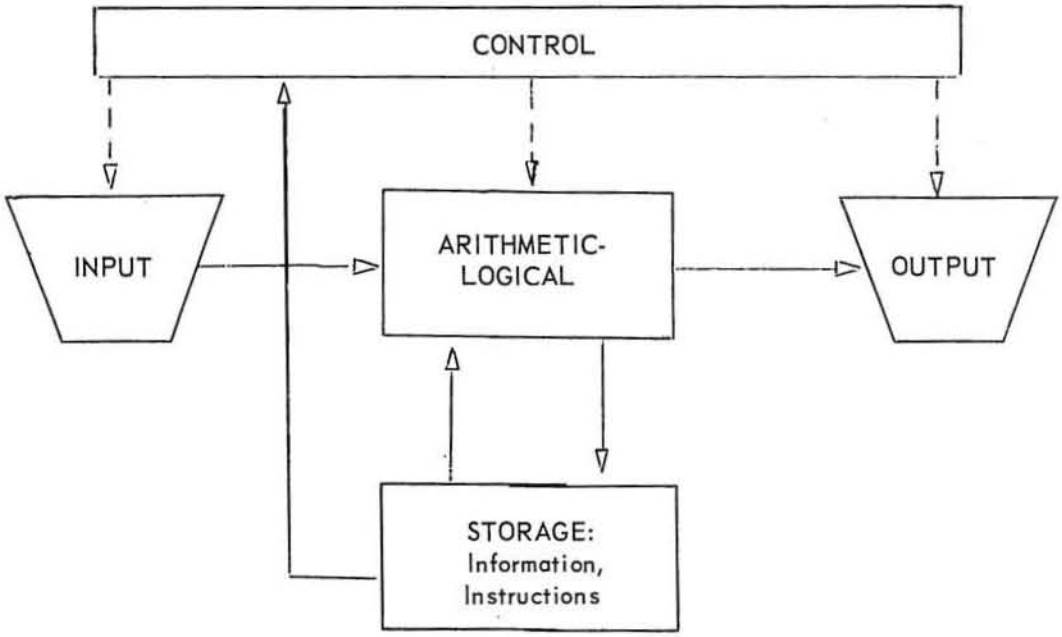
The STORAGE, CONTROL, ARITHMETIC and LOGICAL elements of the Monrobot XI computer are combined into a single unit called the processor, or electronics unit. Included in the processor is a power supply, all the necessary circuitry to perform the control, arithmetic and logical operations, and the magnetic drum used for storage of information and program instructions. The console panel on top of the processor provides for operator control of the computer.



LOGICAL. The computer can distinguish between positive, negative and zero values and can make decisions which relate to these conditions. This is called computer logic.

OUTPUT. The information which is the result of computation may be printed on the typewriter or punched into a paper tape.

The organization of these elements to form the Monrobot XI computer is diagrammed below:



THE STORED PROGRAM

A computer program consists of a sequence of instructions and numeric values used as constants. The program must define every detail of operation and provide control for every solution of a possible set of circumstances. The instructions for solution of a problem are stored in the computer in the sequence required to solve the problem. The computer may refer to them in sequence, or repeat groups of related instructions, may omit or alter instructions, or make logical decisions depending upon anticipated circumstances.

Each instruction, sent in its turn to the control unit, will tell the computer what operation to perform, where to find the information on which to perform it, or what to do with the result.

SUMMARY

The elements of a computer consist of input and output units and a processor which provides control, arithmetic and logical operations, and storage of information and program instructions. Under control of program instructions information is input to the computer, computations are made, and the results saved in storage for subsequent use or sent to an output unit. This is called a data processing system.

To create a data processing system it is first necessary to define the problem in detail, then to translate this detail into a procedure language. This is called programming. This manual describes the Quikomp Programming System. Quikomp is a decimally coded procedure language which simplifies translation of problem details into a computer solution.

INTRODUCTION TO THE QUIKOMP PROGRAM SYSTEM

The Quikomp Programming System was designed by Monroe International to provide a rapid, simple means of using the Monrobot XI computer as a mathematical tool.

The Quikomp Programming System requires a Monrobot XI computer with one input/output typewriter and a paper tape reader and paper tape punch. This is called hardware.

A special set of instructions called the Quikomp Program System causes the computer to operate as described in this manual. These instructions are contained on a punched paper tape and must be entered into the computer in order to operate the Quikomp Program System. This is called software.

This manual describes how to write programs using the Quikomp System and how to operate the Monrobot XI computer. No prior computer experience is required.

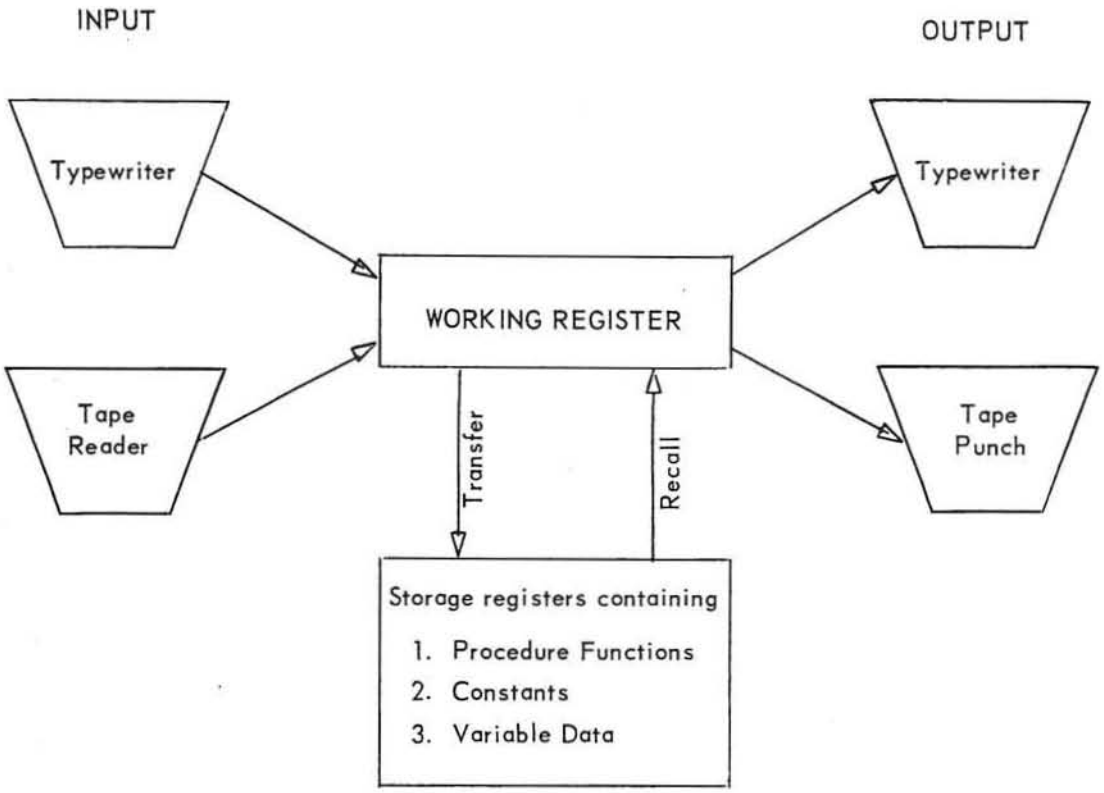
THE MONROBOT XI QUIKOMP COMPUTER

The Quikomp System uses a typewriter or a punched paper tape reader for input of digital information. The typewriter and a paper tape punch provide for output of digital information.

The working register of the Quikomp System performs all arithmetic and mathematical functions. Input and output of digital information goes to and from the working register. Information can be transferred from the working register to storage registers or can be recalled to the working register from storage registers.

The storage registers of the Quikomp System can contain either one procedure function or one numeric data value. Each register has a decimal identification called an address. Register addresses begin with the number 1. There are 320 register addresses in the Quikomp System.

ORGANIZATION OF THE QUIKOMP COMPUTER



SOME DEFINITIONS

A PROGRAM is a sequence of events or functions which occur one after the other. A computer program is a sequence of predetermined functions which lead to the solution of a problem.

A PROCEDURE FUNCTION is a computer action. Input numbers, add, and find square root are examples of computer procedure functions.

A FUNCTION CODE is a decimal number for a procedure which the computer will interpret as meaning input numbers, add, or find square root.

AN OPERAND identifies a location which is to be used when performing a procedure function. An operand is a register address which answers the question "what?" or "where?"

THE PROGRAM SHEET

Each line of the program sheet is used to describe an action or function which the computer is to perform. Part of the description is used by the programmer and part will be used by the computer.

PROCEDURE DESCRIPTION consists of English words and/or mathematical symbols which define what is to occur. This description is the logical planning of a program. It will not be entered into the computer, it is for the programmer's information only.

REGISTER is a location address in the computer. A storage register address may contain a code to cause the computer to perform a function or an address may contain a data value upon which a function will be performed. Registers are identified by decimal addresses from 1 to 320.

CODE is a decimal number which stands for a Quikomp program function. The functions in the Quikomp Program System and the code assigned to each is contained in the body of this manual. The function code is entered into the computer.

OPERAND is a register address indicating "where" or "with what" a program function is to be performed. The operand address is entered into the computer.

MONROBOT XI QUIKOMP™ PROGRAM SHEET		PAGE OF
PROGRAM	PROGRAMMER	DATE / /

PROCEDURE DESCRIPTION	REGISTER	CODE	OPERAND
	1		
	2		
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	0		
	1		
	2		
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	0		
	1		
	2		
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	0		

PREPARATION OF A PROGRAM

A description of the procedure required for the solution of a mathematical problem is prepared first. The procedure description is written on the Quikomp Program Sheet to clarify the logical steps required to solve a problem. Each line of the procedure description will become a single Quikomp program function as described in this manual. There are no rules concerning procedure description; it is correct if it defines the problem. Writing of the procedure description is called programming.

After the procedure description has been completed Quikomp function codes are assigned for each register (program sheet line). Many function codes require an operand to identify the "what" and "where." Most operands will be the address of another register location where it is planned to store numeric data.

Quikomp function codes and operands are always decimal numbers. The process of assigning function codes and operands is called coding.

QUIKOMP PROGRAM FUNCTION DESCRIPTIONS

GENERAL

The Quikomp program functions described in this manual are divided into five groups and will be discussed in this order:

1. Numeric Input Functions
2. Numeric Output Functions
3. Data Transfer Functions
4. Program Control Functions
5. Arithmetic Functions
6. Mathematical Functions

FORMAT

Each of the Quikomp functions will be described in the following format:

Function	- The English name of the function.
Code	- A numeric code which stands for the function.
Operand	- A register address or special code with which the function will operate.
Description	- An explanation of the function and what it causes the computer to do.
Example	- An illustration of the working register and storage registers before and after operation of a program function.

omit see page 42

NUMERIC INPUT FUNCTIONS

Arithmetic and mathematical computations are performed on decimal numbers in the computer. Before such calculations can take place all necessary numeric values must be entered into the computer. This is called numeric input.

The Quikomp input functions allow entry of numeric data into the working register of the computer. Data may be input from the typewriter or from a punched paper tape and the tape reader.

Input of numeric data places the new input data in the working register. The previous contents of the working register will be lost.

The digits 1, 2, 3, 4, 5, 6, 7, 8, 9 and zero are used for input of numeric data. A decimal point (.) and a minus sign (-) are used when required.

QUIKOMP NUMERIC SYSTEM

Numeric data values can contain up to seven significant or real digits. A decimal point can be included to express the number properly. A minus sign must be entered when the data value is intended to be negative. Some examples of correct numeric data are:

5.9

4673.98

-465

.000386

-.05

998.6651

Numeric data values can have a wide range of magnitude. This is called scaling. Up to 63 places to the left or right of the decimal place are possible in the Quikomp numeric system. Within these 63 places there will be seven significant digits preceded or followed by the correct number of zeros to indicate the scaling of the number. Some examples of numbers of large and small magnitude are:

8856639000000000

1234567000

-9935678

-.000004467398

If numeric data of more than seven significant digits is input to the computer, rounding may occur in the seventh place. Rounding takes place when the eighth digit is 5 or greater. The correct scaling is maintained but the value will be truncated or reduced to the seven most significant digits. If the value

3.14159265358979

is input to the computer the result will be rounded and truncated so that the working register will contain

3.141593

INPUT FUNCTIONS

Function	Read typewriter
Code	1
Operand	Not required

DESCRIPTION

The *read typewriter* function permits input of numeric data from the typewriter keyboard. After input the numeric data is contained in the working register. The previous contents of the working register will be lost.

EXAMPLE:

Read typewriter

Function code: 1

	WORKING REGISTER
Assuming present contents are	3.456
After input from typewriter	100.754
Present contents	.00345
After input	-3.1415

Function	Read punched tape
Code	3
Operand	Not required

DESCRIPTION

The *read punched tape* function permits input of numeric data from the punched paper tape reader. After input the numeric data is contained in the working register. The previous contents of the working register will be lost.

EXAMPLE:

Read punched tape

Function code: 3

	WORKING REGISTER
Assuming present contents are	99.7
After input from tape reader	.00034
Present contents	-1
After input	500000

NUMERIC OUTPUT FUNCTIONS

The results of arithmetic or mathematical computations can be printed on the typewriter or punched into paper tape. Printed results are the record of problem solution. The results of computations can also be punched into paper tape if further calculations are to be performed on the numeric data.

The Quikomp output functions allow output of numeric data from the working register to the computer. Data may be output to the typewriter, or to the tape punch, or to the typewriter and the tape punch in combination.

Output of numeric data does not affect the contents of the working register. The numeric value remains in the working register after the output function takes place.

Output of numeric data is printed in the same manner as described for input data.

Other Quikomp output functions are used to control the format of printed results or print alphabetic identifying information.

OUTPUT FUNCTIONS

Function	Write typewriter
Code	5
Operand	Not required

DESCRIPTION

The *write typewriter* function prints the contents of the working register on the typewriter. The contents of the working register are not affected.

The numeric output value will be printed with minimum of seven digits and will contain a decimal point and/or a minus sign if required.

EXAMPLE:

Write typewriter

Function code: 5

	WORKING REGISTER	PRINTED AS
Assuming present contents are	1.	
After output to typewriter	1.	1.000000
Present contents	- 12.	
After output	- 12.	-12.00000
Present contents	.00012345	
After output	.00012345	.0001234500

Function	Write tape
Code	7
Operand	Not required

DESCRIPTION

The *write tape* function punches the contents of the working register on the paper tape punch. The contents of the working register are not affected.

The numeric value will be punched with a minimum of seven digits and will contain a decimal point and/or minus sign if required. An additional character, called an end code, will be punched immediately following the numeric value. This code is used to separate numeric data on a punched tape.

EXAMPLE:

Write tape

Function code: 7

	WORKING REGISTER	PUNCHED AS
Assuming present contents are	1457.0	
After output to tape punch	1457.0	1457.000
Present contents	- .5	
After output	- .5	- .5000000

Function	Write typewriter and tape
Code	9
Operand	Not required

DESCRIPTION

The *write typewriter and tape* function simultaneously prints the contents of the working register on the typewriter and punches the same information on the paper tape punch. The contents of the working register are not affected.

The numeric value will be printed and punched with a minimum of seven digits and will contain a decimal point and/or a minus sign if required.

EXAMPLE:

See *Write typewriter* and *Write tape*.

Function	Carriage return typewriter
Code	11
Operand	Not required

DESCRIPTION

The *carriage return typewriter* function causes the typewriter carriage to return to the left margin and advance the paper to the next line. The contents of the working register are not affected.

Function	Tab typewriter
Code	13
Operand	Not required

DESCRIPTION

The *tab typewriter* function causes the typewriter carriage to move to the right on the same printing line. The carriage will stop at the next available tab stop setting. The tab function permits the printing of numeric data in columnar form. The contents of the working register are not affected.

Function	Carriage return, form-up page
Code	15
Operand	Not required

DESCRIPTION

The *carriage return, form-up* function can be used in place of *carriage return,* function code 11, when more than one page of data is to be printed. The form-up function permits printing of 60 lines after which 6 lines will be skipped.

Continuous form paper which is perforated every eleven inches is used. At the start of program operation the paper in the typewriter must be adjusted so that the first printout occurs on the second or third line below the perforation.

The contents of the working register are not affected.

Function	Write character
Code	2
Operand	See Table 1

DESCRIPTION

The *write character* function will cause the typewriter to print a single character corresponding to the code in the operand. The contents of the working register are not affected.

EXAMPLE:

Write character

Function code: 2 49

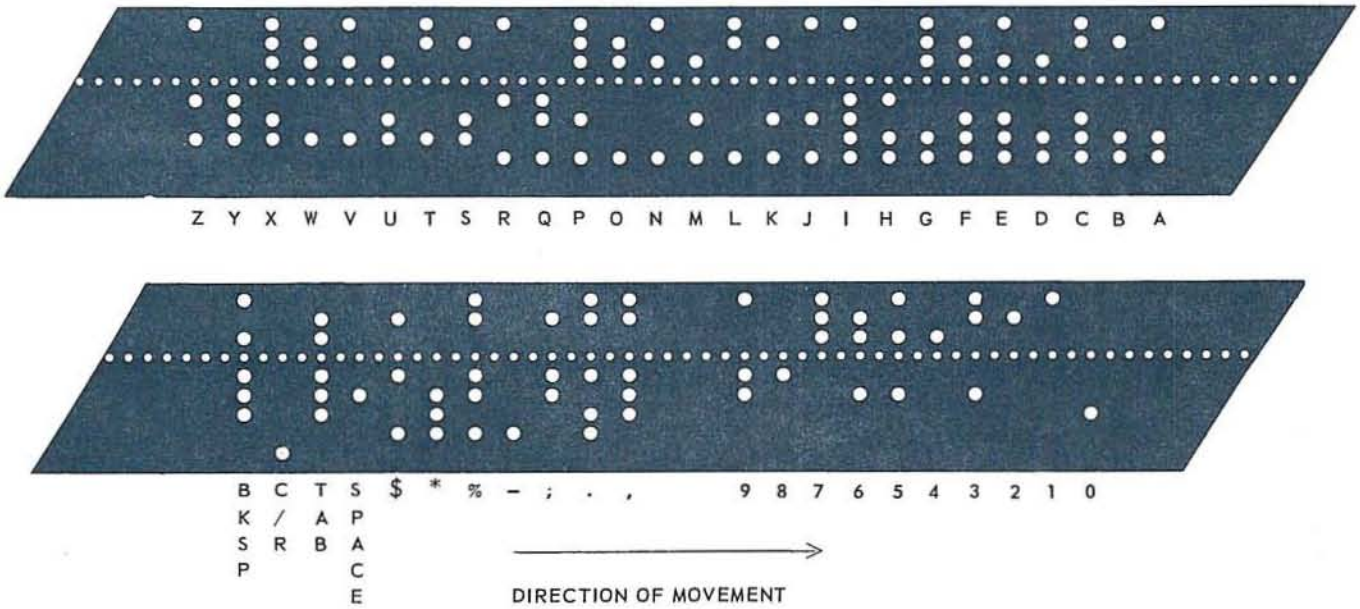
	WORKING REGISTER	PRINTED AS
Assuming present contents are	33.45	
After output to typewriter	33.45	A

Function	Punch character
Code	4
Operand	See Table 1

DESCRIPTION

The *punch character* function will cause the tape punch to punch a single character corresponding to the code in the operand. The contents of the working register are not affected.

EXAMPLE:



Function Write and punch character

Code 6

Operand See Table 1

DESCRIPTION

The *write and punch character* function will cause simultaneous printing and punching of a single character corresponding to the code in the operand. The contents of the working register are not affected.

EXAMPLE:

See *Write character* and *Punch character.*

TABLE 1

OUTPUT CHARACTER CODES

CHAR.	OPERAND	CHAR.	OPERAND
A	49	0	16
B	50	1	1
C	115	2	2
D	52	3	67
E	117	4	4
F	118	5	69
G	55	6	70
H	56	7	7
I	121	8	8
J	97	9	73
K	98		
L	35		
M	100		
N	37	,	91
O	38	.	59
P	103	;	74
Q	104	-	32
R	41	%	107
S	82	*	112
T	19	$	42
U	84	Space	64
V	21	Tab	94
W	22	C/R	128
X	87	Bksp	93
Y	88	U Case	124
Z	25	L Case	122

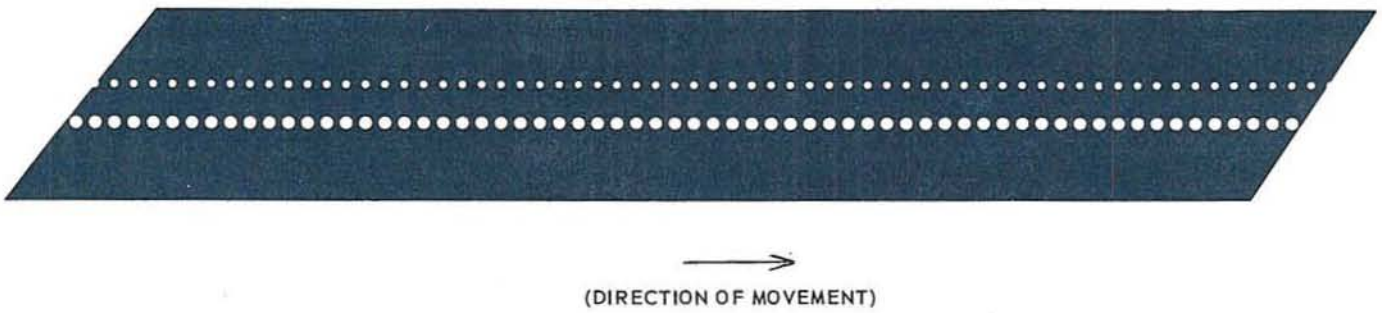
Function	Punch tape feed
Code	17
Operand	Not required

DESCRIPTION

The *punch tape feed* function causes output of twelve inches of tape feed to the tape punch. The contents of the working register are not affected.

Tape feed is required to precede and follow the first and last numeric data which is punched into paper tape. This will permit proper threading of the tape in the tape reader.

EXAMPLE OF TAPE FEED:



DATA TRANSFER

The Quikomp data transfer functions permit the movement of data values between the working register and storage registers.

A copy of the contents of the working register can be transferred to a storage register, or a copy of the contents of a storage register can be recalled to the working register.

DATA TRANSFER FUNCTIONS

Function	Recall
Code	8
Operand	Register address

DESCRIPTION

The *recall* function brings a copy of the contents of the operand register to the working register. The previous contents of the working register are lost, the operand register is not affected.

EXAMPLE:

Recall

Function code: 8

	WORKING REGISTER	OPERAND REGISTER
Assuming present contents are	2.35	45.39
After recall	45.39	45.39
Present contents	566.978	5
After recall	5	5

Function	Transfer
Code	10
Operand	Register Address

DESCRIPTION

The *transfer function* places a copy of the working register in the operand register. The previous contents of the operand register are lost, the working register is not affected.

EXAMPLE:

Transfer

Function code: 10

	WORKING REGISTER	OPERAND REGISTER
Assuming present contents are	55.2	678.32
After transfer	55.2	55.2
Present contents	46	65843.78
After transfer	46	46

Function	Interchange
Code	38
Operand	Register address

DESCRIPTION

The *interchange* function exchanges the contents of the operand register with the contents of the working register. No information is lost.

EXAMPLE:

Interchange
Function code: 38

	WORKING REGISTER	OPERAND REGISTER
Assuming present contents are	8.889	3667.9
After interchange	3667.9	8.889
Present contents	45	5674.48
After interchange	5674.48	45

Function	Clear working register
Code	23
Operand	Not required

DESCRIPTION

The *clear working register* function sets the contents of the working register to zero. The previous contents of the working register are lost, no other register is affected.

EXAMPLE:

Clear working register

Function code: 23

	WORKING REGISTER
Assuming present contents are	456.7563
After operation	0.
Present contents	-5.75
After operation	0.

Function	Clear register
Code	36
Operand	Register address

DESCRIPTION

The *clear register* function sets the contents of the operand register to zero. The previous contents of the operand register are lost, no other register is affected.

EXAMPLE:

Clear register

Function code 36

	WORKING REGISTER	OPERAND REGISTER
Assuming present contents are	453.6	75649
After operation	453.6	0.
Present contents	768	4
After operation	768	0.

PROGRAM CONTROL

A computer program consists of a series of functions which are normally performed in sequence. However, it is sometimes necessary to alter this sequence. The program functions which alter the course of a computer program are classed as unconditional or conditional.

Unconditional functions do not involve a decision, they are mandatory. The unconditional program control functions in the Quikomp programming system are:

Go to

Subroutine entry

Subroutine exit

Pause

No operation

End of program

Conditional functions involve a logical decision based on the contents of the working register. The result of the decision is 1) change program sequence, or 2) continue the program sequence. The conditional program control functions are:

If negative go to

If zero go to

Sense switch

A special group of program control functions cause the program to repeat a sequence of functions a fixed number of times. They are called looping functions and are:

Set loop counter

Modify next operand

Loop return

The loop functions are a form of conditional program control.

SUBROUTINES AND LOOPS

Often during a program it is necessary to repreat identical sequences of functions. If these functions are to be performed at different points in a program the sequence is written as a subroutine. If the sequence of functions is to be performed repeatedly at one point in the program it is written as a loop. Subroutines and loops simplify the writing of a program and conserve storage registers.

A subroutine is a sequence of program functions set apart from the main program. The subroutine entry function written in the main program directs the computer to the dependent program. The subroutine exit function written at the end of the dependent program directs the computer back to the main program. A subroutine is used two or more times by the main program.

A loop is a sequence of functions written in the main program. The loop is performed a fixed number of times (two or more) before the main program is continued. Within a loop it is possible to perform a function on one register during the first loop, then the next register during the second loop, etc. This is called register address modification.

RULES FOR SUBROUTINES AND LOOPS

1. A program may contain any number of subroutines.

    a. Each subroutine must be completed before the next subroutine can begin.

2. A program may contain any number of loop sequences.

    a. Each loop sequence must be completed before the next loop can begin.

3. A subroutine can contain one or more loops observing the conditions of rule 2, a.

4. A loop can contain one or more subroutines observing the conditions of rule 1, a.

Function Go to

Code 12

Operand Register address

DESCRIPTION

The *go to* function causes the computer to begin a new sequence of program functions starting with the function contained in the operand register address. The contents of the working register are not affected.

EXAMPLE:

Go to

Function code 12

REG.	CODE	OPERAND	WORKING REGISTER	NEXT REGISTER
49	12	93	not affected	93

Function If negative go to

Code 14

Operand Register address

DESCRIPTION

The *if negative go to* function tests the condition of the working register. If a negative data value is present in the working register the computer will begin a new sequence of program functions starting with the function contained in the operand register address.

If the value in the working register is not negative the computer will continue to the register address following the *if negative* function.

The contents of the working register are not affected.

EXAMPLE:

If negative go to

Function code 14

REG.	CODE	OPERAND	WORKING REGISTER	NEXT REGISTER
72	14	150	-1.5	150
72	14	150	.0159	73

Function If zero go to

Code 16

Operand Register address

DESCRIPTION

The *if zero go to* function tests the condition of the working register. If a zero data value is present in the working register the computer will begin a new sequence of program functions starting with the function contained in the operand register address.

If the value in the working register is not zero the computer will continue to the register address following the *if zero* function.

The contents of the working register are not affected.

EXAMPLE:

If zero go to

Function code 16

REG.	CODE	OPERAND	WORKING REGISTER	NEXT REGISTER
22	16	39	0.	39
22	16	39	172.3	23

Function Subroutine entry

Operand Register address

DESCRIPTION

The *subroutine entry* function causes the computer to begin a new sequence of program functions starting with the function contained in the operand register address. The contents of the working register are not affected.

Subroutine entry is a special form of the *go to* function which causes the computer to retain the address of the register which follows. After the functions of the subprogram are performed the *subroutine exit* function will return the computer to the register address retained by the *subroutine entry* function.

Function Subroutine exit

Code 19

Operand Not required

DESCRIPTION

The *subroutine exit* function causes the computer to return to the register address retained by the last use of the *subroutine entry* function. The *subroutine exit* function must always be used to end a subprogram. The contents of the working register are not affected.

EXAMPLE:

Subroutine entry

Function code 18

REG.	CODE	OPERAND	WORKING REGISTER	NEXT REGISTER
21	18	105	Not affected	105
22	(continuation of program after subroutine exit)			115

EXAMPLE:

Subroutine exit

Function code 19

REG.	CODE	OPERAND	WORKING REGISTER	NEXT REGISTER
115	19	-	Not affected	22

Function No Operation

Code 21

Operand Not required

DESCRIPTION

The *no operation* function is a "do nothing" operation and causes no action by the computer. The computer will sequence to the next register address automatically. The contents of the working register are not affected.

The *no operation* function is used for program corrections or alterations. It can be substituted for a function which is no longer required in a program. If program alterations are planned the *no operation* function can be used to fill a register which is to be changed after preliminary testing of the program.

EXAMPLE:

No operation

Function code 21

REG.	CODE	OPERAND	WORKING REGISTER	NEXT REGISTER
56	21	-	Not affected	57

Function Pause

Code 22

Operand 1 to 9

DESCRIPTION

The *pause* function causes the computer to stop processing. The operand number is displayed in the control register lights on the computer console panel. Processing can be resumed by pressing the START button. The contents of the working register are not affected.

OPERAND NUMBER	CONTROL REGISTER DISPLAY*
1	0000 0001 0000 0001
2	0000 0001 0000 0010
3	0000 0001 0000 0011
4	0000 0001 0000 0100
5	0000 0001 0000 0101
6	0000 0001 0000 0110
7	0000 0001 0000 0111
8	0000 0001 0000 1000
9	0000 0001 0000 1001

* 0= not lighted; 1= lighted

EXAMPLE:

Pause

Function code 22

REG.	CODE	OPERAND	WORKING REGISTER	NEXT REGISTER
82	22	3	Not affected	83 (after pressing START button)

Function	Sense Switch	
Code	20	
Operand	1 for switch 1	16 for switch 5
	2 for switch 2	32 for switch 6
	4 for switch 3	64 for switch 7
	8 for switch 4	128 for switch 8

DESCRIPTION

The *sense switch* function tests the condition of the switch specified by the operand. If the switch has been manually set OFF, the working register is set to zero. If the switch has been manually set ON, the working register is set to negative. The previous contents of the working register are always lost.

The *sense switch* function must be followed by the *If Negative go to* function or the *If Zero go to* function.

EXAMPLE:

Sense switch

Function code 20

REG.	CODE	OPERAND	WORKING REGISTER	NEXT REGISTER
Switch ON				
21	20	2	set negative	22
22	17	95	negative	95
Switch OFF				
21	20	2	set zero	22
22	17	95	zero	23

Function	Set loop counter
Code	24
Operand	2 to 99 loops

DESCRIPTION

The *set loop counter* function initializes the looping function to the value specified by the operand. The value given in the operand specifies the number of times a loop of program functions is to be performed. The count of loops which have been performed is set to zero by the *set loop counter* function.

The *set loop counter* function must be performed prior to the first use of the *modify next operand* or *loop return* functions. A loop must be completed before the *set loop counter* can be used again to begin another loop.

The contents of the working register are not affected.

Function	Modify next operand
Code	25
Operand	Not required

DESCRIPTION

The *modify next operand* function affects the operand address of the register immediately following. The address of the next operand is modified by addition of the count of the number of loops which have been performed. The contents of the working register are not affected.

The first time the *modify next operand* function is performed the effective address is the one written in the program. That is, a zero count of loops performed is added to the next operand address. Each repetition of a loop will cause a 1, 2, 3, 4, etc., to be added to the next operand address.

This procedure is called address modification and makes it possible for the computer to perform the same function but on the contents of a different register each time through a loop.

The functions which can have an operand address modified are

FUNCTION	CODE
Recall	8
Transfer	10
Interchange	38
Clear register	36
Go to	12
If negative go to	14
If zero go to	16
Add	28
Subtract	30
Multiply	32
Divide	34

Function	Loop return
Code	26
Operand	Register address

DESCRIPTION

The *loop return* function must be performed as the last of the group of functions within a loop. The *loop return* function adds a 1 to the count of loops which have been performed and tests the result against the number of loops which are to be performed. If looping has not been completed the program will return to the function specified by the operand register address. The return register address must be a location following the *set loop counter* function. If looping has been completed the program will continue to function located in the register following the *loop return* function.

The contents of the working register are not affected.

MONROBOT XI QUIKOMP™ PROGRAM SHEET

PAGE OF

PROGRAM: Example of looping

PROGRAMMER:

DATE: / /

PROCEDURE DESCRIPTION	REGISTER	CODE	OPERAND
Set loop counter	101	24	10
Carriage return	102	11	
Input data from typewriter	103	1	
Modify next operand	104	25	
Transfer input data	105	10	250
Loop return	106	26	103
Continuation of program	7		
	8		
	9		
	0		
	1		
	2		
This loop will allow ten data values	3		
to be input from the typewriter and	4		
transferred from the working register	5		
to data storage registers 250 to 259.	6		
	7		
	8		
	9		
	0		
	1		
	2		
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	0		

Function	End of Program
Code	0
Operand	Not required

DESCRIPTION

The *end of program* function causes the computer typewriter to carriage return and transfer control to the six Quikomp control switches. The contents of the working register are not affected. *End of program* must be used as the final function in a program sequence. The contents of the working register are not affected.

EXAMPLE:

End of program

Function code . 0

REG.	CODE	OPERAND	WORKING REGISTER	NEXT REGISTER
249	0	-	Not affected	Quikomp control switches

ARITHMETIC FUNCTIONS

The Quikomp arithmetic functions perform addition, subtraction, multiplication and division. The arithmetic functions operate on the contents of the working register and the contents of a data register address. The result of an arithmetic function is contained in the working register. The previous contents of the working register are lost, the contents of the data register address are unchanged.

Function	Add
Code	28
Operand	Register address

DESCRIPTION

The *add* function finds the sum of the contents of the working register and the contents of the operand register. After addition the previous contents of the working register are lost, the contents of the operand register are unchanged.

EXAMPLE:

Add

Function code: 28

	WORKING REGISTER	OPERAND REGISTER
Assuming present contents are	35.65	7.8
After addition	43.45	7.8
Present contents	2.45	-5.67
After addition	-3.22	-5.67

Function	Subtract
Code	30
Operand	Register address

DESCRIPTION

The *subtract* function find the difference between the contents of the working register and the contents of the operand register. After the contents of the operand register are subtracted from the contents of the working register the difference is contained in the working register. The previous contents of the working register are lost, the contents of the operand register are unchanged.

EXAMPLE:

Subtract

Function code 30

	WORKING REGISTER	OPERAND REGISTER
Assuming present contents are	20.9	5.6
After subtraction	15.3	5.6
Present contents	-25.9	6
After subtraction	-31.9	6

Function	Multiplication
Code	32
Operand	Register address

DESCRIPTION

The *multiplication* function finds the product of the contents of the working register and the contents of the operand register. The product is contained in the working register. After multiplication the previous contents of the working register are lost, the contents of the operand register are unchanged.

EXAMPLE:

Multiply

Function code: 32

	WORKING REGISTER	OPERAND REGISTER
Assuming present contents are	2.0	4.35
After multiplication	8.70	4.35
Present contents	-2.1	3
After multiplication	-6.3	3

Function Division

Code 34

Operand Register address

DESCRIPTION

The *division* function finds the quotient of the contents of the working register and the contents of the operand register. After the contents of the working register are divided into the contents of the operand register the quotient is contained in the working register. The previous contents of the working register are lost, the contents of the operand register are unchanged. Division by zero is illegal.

EXAMPLE:

Divide

Function code: 34

	WORKING REGISTER	OPERAND REGISTER
Assuming present contents are	3.00	12.00
After division	4.00	12.00
Present contents	2.75	-3.1416
After division	-1.1424	-3.1416

MATHEMATICAL FUNCTIONS

The Quikomp mathematical functions are square root and natural or decimal logarithm and antilogarithm. The mathematical functions operate on the value contained in the working register. The result of any mathematical function is contained in the working register, the previous contents of the working register are lost.

Function	Square Root
Code	27
Operand	Not required

DESCRIPTION

The *square root* function finds the square root of the contents of the working register. The square root is contained in the working register, the previous contents of the working register are lost. The square root of a negative number is illegal.

EXAMPLE:

Square Root

Function code: 27

	WORKING REGISTER
Assuming present contents are	9.00
After square root	3.00
Present contents	16
After square root	4

Function	Logarithm, Natural
Code	29
Operand	Not required

DESCRIPTION

The *natural log* function finds the log to the base e of the contents of the working register. The logarithm is contained in the working register. The logarithm is contained in the working register, the previous contents of the working register are lost. The logarithm of zero or a negative number is illegal.

EXAMPLE:

Log, natural

Function code: 29

	WORKING REGISTER
Assuming present contents are	9.40
After log e	2.240709
Present contents	60.20
After log e	4.097672

Function	Logarithm, Decimal
Code	31
Operand	Not required

DESCRIPTION

The *decimal log* function finds the log to the base 10 of the contents of the working register. The logarithm is contained in the working register, the previous contents of the working register are lost. The log of zero or a negative number is illegal.

EXAMPLE:

Log, decimal

Function code: 31

	WORKING REGISTER
Assuming present contents are	1.352
After log 10	.1309766
Present contents	4.03
After log 10	.6053051

Function	Antilogarithm, Natural
Code	33
Operand	Not required

DESCRIPTION

The *natural antilog* function finds the antilog to the base e of the contents of the working register. The antilogarithm is contained in the working register, the previous contents of the working register are lost.

EXAMPLE:

Antilog, natural

Function code: 33

	WORKING REGISTER
Assuming present contents are	1.02962
After antilog e	2.799999
Present contents	4.24850
After antilog e	70.00031

Function	Antilogarithm, Decimal
Code	35
Operand	Not required

DESCRIPTION

The *decimal antilog* function finds the antilog to the base 10 of the contents of the working register. The antilogarithm is contained in the working register, the previous contents of the working register are lost.

EXAMPLE:

Antilog, decimal

Function code: 35

	WORKING REGISTER
Assuming present contents are	.30320
After antilog 10	2.010018
Present contents	.43152
After antilog 10	2.700971

THE QUIKOMP II PROGRAM SYSTEM

The Quikomp II Programming System is designed for use on a 2,048 register Monrobot XI computer with input/output typewriter, paper tape reader and punch. All functions of the Quikomp I system are present with the addition of ten trigonometric functions. The Quikomp II system has 1,023 storage registers available for program functions or data. The Quikomp II numeric system is especially suited to scientific and engineering computations.

QUIKOMP II NUMERIC SYSTEM

Numeric data values are represented in the form of a mantissa and an exponent. This is often called "scientific notation."

The mantissa can contain up to seven significant or "real" digits. The decimal point is assumed to be located to the right of the most significant digit. A minus sign is present when the mantissa is negative.

The exponent or scaling factor is the power of ten by which the mantissa must be multiplied to obtain the true value. The exponent can be a positive or negative power of ten. A positive exponent indicates how many positions the decimal point is to be moved to the right to obtain the true value. A negative exponent indicates how many positions the decimal point is to be moved to the left.

The scale of the mantissa plus its power of ten exponent cannot exceed $\pm 10^{63}$.

Numeric values are written in the form of ± X.XXXXXX/ ± EE where X.XXXXXX is the mantissa or real number and EE is the exponent or scaling factor.

Examples of correct numeric data are:

3.65732/10	which is equal to	36,573,200,000
4.3/2	which is equal to	430
-6.765/3	which is equal to	-6,765
9/-5	which is equal to	.00009

Function	Sine of degrees
Code	39
Operand	Not required

DESCRIPTION

The *sine of degrees* function finds the sine of the contents of the working register. The working register must contain a number expressed in degrees. The sine is contained in the working register, the previous contents of the working register are lost. The sine of a number greater than 999°, or less than -999° is illegal.

EXAMPLE:

Sine, degrees

Function code: 39

	WORKING REGISTER
Assuming present contents are	40.0
After sine	.6427876
Present contents	36.15
After sine	.5899012

Function	Sine of radians
Code	41
Operand	Not required

DESCRIPTION

The *sine of radians* function finds the sine of the contents of the working register. The working register must contain a number expressed in radians. The sine is contained in the working register, the previous contents of the working register are lost. The sine of a number greater than 21.4 radians or less than -21.4 radians is illegal.

EXAMPLE:

Sine, radians

Function code: 41

	WORKING REGISTER
Assuming present contents are	1.50
After sine	.9974950
Present contents	.14
After sine	.1395431

Function	Cosine of degrees
Code	43
Operand	Not required

DESCRIPTION

The *cosine of degrees* function finds the cosine of the contents of the working register. The working register must contain a number expressed in degrees. The cosine is contained in the working register, the previous contents of the working register are lost. The cosine of a number greater than 999°, or less than -909° is illegal.

EXAMPLE:

Cosine, degrees

Function code: 43

Assuming present contents are	37.0
After cosine	.7986355
Present contents	24.50
After cosine	.9099613

Function	Cosine of radians
Code	45
Operand	Not required

DESCRIPTION

The *cosine of radians* function finds the cosine of the contents of the working register. The working register must contain a number expressed in radians. The cosine is contained in the working register, the previous contents of the working register are lost. The cosine of a number greater than 21.4 radians or less than -21.4 radians is illegal.

EXAMPLE

Cosine, radians

Function code: 45

Assuming present contents are	.35
After cosine	.9393727
Present contents	1.65
After cosine	-.07912090

Function	Tangent of degrees
Code	47
Operand	Not required

DESCRIPTION

The *tangent of degrees* function finds the tangent of the contents of the working register. The working register must contain a number expressed in degrees. The tangent is contained in the working register, the previous contents of the working register are lost. The tangent of a number greater than 909°, or less than -909°, or the tangent of 90° is illegal.

EXAMPLE:

Tangent, degrees

Function code: 47

	WORKING REGISTER
Assuming present contents are	42.0
After tangent	.9004041
Present contents	36.50
After tangent	.7399611

Function	Tangent of radians
Code	49
Operand	Not required

DESCRIPTION

The *tangent of radians* function finds the tangent of the contents of the working register. The working register must contain a number expressed in radians. The tangent is contained in the working register, the previous contents of the working register are lost. The tangent of a number greater than 21.4 radians or less than -21.4 radians is illegal.

EXAMPLE:

Tangent, radians

Function code: 49

	WORKING REGISTER
Assuming present contents	.11
After tangent	.1104458
Present contents	1.71
After tangent	-7.137254

Function	Cotangent of degrees
Code	51
Operand	Not required

DESCRIPTION

The *contangent of degrees* function finds the cotangent of the contents of the working register. The working register must contain a number expressed in degrees. The cotangent is contained in the working register, the previous contents of the working register are lost. The cotangent of a number greater than 909°, or less than -909°, or the cotangent of zero is illegal.

EXAMPLE:

Cotangent, degrees

Function code: 51

	WORKING REGISTER
Assuming present contents are	43.0
After cotangent	1.072369
Present contents	36.50
After cotangent	1.351423

Function	Cotangent of radians
Code	53
Operand	Not required

DESCRIPTION

The *cotangent of radians* function finds the cotangent of the contents of the working register. The working register must contain a number expressed in radians. The cotangent is contained in the working register, the previous contents of the working register are lost. The cotangent of a number greater than 21.4 radians or less than -21.4 radians, or the cotangent of zero is illegal.

EXAMPLE:

Cotangent, radians

Function code: 53

	WORKING REGISTER
Assuming present contents are	.18
After cotangent	5.495425
Present contents	1.57
After cotangent	.0007963002

Function	Arctangent
Code	55
Operand	Not required

DESCRIPTION

The *arctangent* function finds the arctangent of the contents of the working register. The arctangent expressed in radians is contained in the working register, the previous contents of the working register are lost.

EXAMPLE:

Arctangent

Function code: 55

	WORKING REGISTER
Assuming present contents are	.14092
After arctangent	.1399979
Present contents	-3.4881
After arctangent	-1.291596

Function	Arccotangent
Code	57
Operand	Not required

DESCRIPTION

The *arccotangent* function finds the arccotangent of the contents of the working register. The arccotangent expressed in radians is contained in the working register, the previous contents of the working register are lost.

EXAMPLE:

Arccotangent

Function code: 57

	WORKING REGISTER
Assuming present contents are	24.987
After arccotangent	.03999932
Present contents	-.23330
After arccotangent	-1.341596

OPERATION OF THE QUIKOMP COMPUTER

The operating descriptions which follow will assume some basic knowledge on the part of the user. This pertains primarily to the physical operation of the computer rather than technical programming concepts. Persons having little or no experience in operation of the Monrobot XI computer should turn immediately to Appendices I, II and III and study them carefully in order to use this portion of the manual.

INPUT OF THE QUIKOMP PROGRAM SYSTEM

The computer must be set up to contain the Quikomp Program System before any operations can take place. To do this, insert and center continuous form paper (8½ x 11") in the formaliner and typewriter carriage. Adjust the left margin and clear all tab stops. Then set new tab stops every 12 spaces (every 15 spaces when using the Quikomp II System).

Place the tape labeled Quikomp I (or II) on the tape reader. Press the RESET button, type 37X03008, then press the LOAD and START buttons. The Quikomp Program System will be read into the computer. After the program is input the START button will remain lighted and the control register lights will continue to flash. Review the program input procedures in appendix III if further instructions are needed.

RESTART OF THE QUIKOMP SYSTEM

If an operation must be interrupted before it is completed, press the RESET button; type 30080000, then press the LOAD and START buttons. The computer will scan the Quikomp control switches and a selection can be made.

QUIKOMP CONTROL SWITCHES

Operation of the Quikomp Programming System is controlled by six sense switches. The use of these switches is as follows:

Switch 1 - Learn

A Quikomp program can be input to the computer from the console typewriter keyboard.

Switch 2 - Correct

A Quikomp program which has been input to the computer can be corrected from the typewriter keyboard. Single registers or a sequence of registers can be changed.

Switch 3 - Execute

This switch is used to begin operation of a Quikomp program which has input to the computer.

Switch 4 - Data Preparation

Numeric data which is to be input from the tape reader during operation of a Quikomp program must be prepared on the computer.

Switch 5 - Punch Program Tape

An operating Quikomp program can be punched into paper tape in compact form for future use. The program tape is verified after punching to insure accuracy.

Switch 6 - Program Input from Add-punch Tape

A program tape can be prepared on the Synchro-Monroe add-punch for input to the computer. This method will save time and avoid typing errors.

At the completion of each operation the computer will return to the Quikomp control switches. The computer will scan the six switches until another operation is selected.

The operating instructions which follow assume that the computer is scanning the Quikomp control switches.

Error control procedures are described in the section which follows the operating instructions.

OPERATING INSTRUCTIONS

SWITCH 1 - LEARN

1. Press switch 1 once to turn it on, and again to turn it off.

2. The computer will print REG. Type the first register address of the program and touch the carriage return key.

3. The computer will print the register address at the left margin. Type the function code for that register, then touch the tab key. Type the operand and touch the carriage return key. The computer will print the next register address. Enter the program line by line in the same manner.

4. If an operand is not required the computer will print a dash after the function code has been entered.

5. The *end of program* (function code zero) will cause the computer to print END in the operand column and return to the Quikomp control switches.

6. If a typing error is made before the tab or carriage return key is used, type an X. The computer will print an E. Re-enter the function code or operand and continue.

SWITCH 2 - CORRECT

To correct a sequence of registers:

1. Press switch 2 once to turn it on and again to turn it off.

2. The computer will print REG. Type the beginning register address of the sequence and touch the carriage return key. The computer will print the register address. Enter the function codes and operands from the typewriter as described for switch 1, learn.

3. The *end of program* function will cause the computer to print END in the operand column and return to the Quikomp control switches.

4. If the *end of program* function is not to be entered press switch 2 to turn it on before entering the last register to be corrected. The computer will again print REG. Press switch 2 to turn it off. Enter a zero register address and touch the carriage return key. The computer will return to the Quikomp control switches.

To correct individual registers:

1. Press switch 2 once to turn it on and leave it on.

2. The computer will print REG. Type the register address to be corrected and touch the carriage return key. The computer will print the register address. Enter the function code and operand from the typewriter as described for switch 1, learn.

3. The computer will again print REG. Repeat step 2 until all registers have been corrected.

4. After all registers have been corrected press switch 2 to turn it off. Enter a zero register address and touch the carriage return key. The computer will return to the Quikomp control switches.

SWITCH 3 - EXECUTE

1. Press switch 3 once to turn it on and again to turn it off.

2. The computer will print REG. Type the starting register address of the program and touch the carriage return key.

3. Program operation will begin.

SWITCH 4 - DATA PREPARATION

1. Press switch 4 once to turn it on and leave it on.

2. Twelve inches of tape feed will be punched on the tape punch.

3. Type the first numeric data value following the switch 3 instructions for Quikomp I or II numeric input. The computer will print the data for verification. If the data is correct go to step 4. If the data is incorrect go to step 5.

4. Touch the space bar. The data will be punched into paper tape. Repeat step 3 to enter all values.

5. Touch the backspace key. Re-enter the correct value. If correct go to step 4; if incorrect repeat this step.

6. Before the last value is accepted (space bar), press switch 4 to turn it off. After the last value is accepted twelve inches of tape feed will be punched and the computer will return to the Quikomp control switches.

SWITCH 5 - PUNCH PROGRAM TAPE

For Quikomp I:

1. Press switch 5 once to turn it on and again to turn it off.

2. The computer will print LOAD TAPE. Place the tape labeled PRGM PCH SW 5 on the reader and press START. The program punch procedure will be read into the computer.

3. The computer will print S REG. Type the minimum register address of the program and touch the carriage return key. The computer will print E REG. Type the maximum register address of the program and touch the carriage return key. The Quikomp program will then be punched into paper tape. (Note: registers which are used for storage of data do not have to be punched in the program tape.)

4. The computer will print PLACE TAPE ON READER. Remove the tape which has just been punched, place it on the tape reader and press START. The program tape will be read for verification. If the tape is incorrect the computer will print ERROR. MAKE A NEW TAPE. Repeat from step 3.

5. If the program tape is correct the computer will print TAPE IS OK. LOAD RESTORE TP. Place the tape labeled Q I RESTORE on the tape reader and press START. The normal Quikomp System will be restored and the computer will return to the Quikomp control switches.

FOR QUIKOMP II:

1. Press switch 5 once to turn it on and again to turn it off.

2. The computer will print S REG. Type the minimum register address of the program and touch the carriage return key. The computer will print E REG. Type the maximum register address of the program and touch the carriage return key. The Quikomp program will then be punched into paper tape. (Note: registers which are used for storage of data do not have to be punched in the program tape.)

3. The computer will print PLACE TAPE ON READER. Remove the tape which has just been punched, place it on the tape reader and press START.

4. The program tape will be read for verification. If the tape is incorrect the computer will print ERROR. MAKE A NEW TAPE. Repeat from step 2.

5. If the program tape is correct the computer will print TAPE IS OK and the computer will return to the Quikomp control switches.

TO INPUT A PROGRAM TAPE PRODUCED BY QUIKOMP SWITCH 5:

1. Press the RESET button and place the program tape on the tape reader.

2. Type 37X03008 and press the LOAD and START buttons.

3. The program tape will be read in and the computer will return to the Quikomp control switches.

SWITCH 6 - PROGRAM INPUT FROM ADD-PUNCH TAPE

Punching the Program Tape:

Persons who are not familiar with the basic operation of the Synchro-Monroe add-punch should refer to the Monroe booklet MO-397.

1. Tear off about seven inches of the listing paper and fold it in half lengthwise twice so that it is about ½" wide. Place it vertically between the 4th and 5th columns of keys. The two columns to the left of the guide paper are used to enter the function codes. The four columns to the right of the paper are used to enter the operands.

2. Set the Vari-field selector at 6, turn the tape punch switch on. Turn the total and sub-total switches off.

3. Press the tape feed button to produce about twelve inches of tape feed.

4. Set the minimum register address of the program in the right-hand side of the keyboard. Touch the plus bar (+) to cause printing and punching.

5. Enter the first function code from the program sheet in the two columns to the left of the guide paper. Enter the operand in the right-hand columns. If an operand is not required make no entry in the right-hand columns. Touch the plus bar to punch the function code and operand. Continue punching the entire program in the same manner.

6. Touch the plus bar without making an entry on the keyboard to enter the "end of program" function. Press the tape feed button to produce twelve inches of tape feed. Remove the punched tape and the listing tape.

   Note: A four-column keyboard cover with a card insert summary of these instructions (MO-315) is available. The cover is placed over the four unused left side columns.

INPUT OF THE PROGRAM TAPE:

1. Place the add-punch program tape on the tape reader.

2. Press switch 6 once to turn it on and again to turn it off.

3. As the program tape is being entered into the computer the register addresses, function codes and operands will be printed on the typewriter. When the "end of program" function is reached END will be printed in the operand column and the computer will return to the Quikomp control switches.

4. If an illegal function code or an operand error is detected the computer will print ERROR. The Quikomp system will substitute a "no operation" function for that register and continue. Corrections are made by using switch 2.

PUNCHING ERRORS:

1. If the starting address was not the first entry in the program tape the computer will print OVRFL ERR. Make a short tape consisting of tape feed, starting address, tape feed. Re-start the Quikomp System, go to step 2 above, use the short address tape followed by the program tape.

2. If the "end of program" function was omitted the computer will read through the ending tape feed and stop; END will not be printed. Make a short tape consisting of tape feed, a zero entry, tape feed. Place this tape in the tape reader to complete program input.

SUMMARY OF ADD-PUNCH INSTRUCTIONS

SWITCH SETTINGS:

Vari-field:	6
Total:	Off
Sub-total:	Off
Tape punch:	ON

Add-Punch Keyboard Columns:

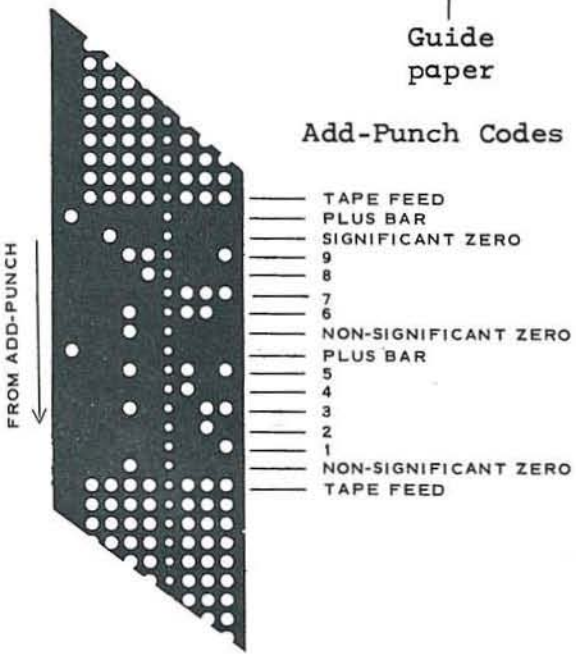
	10	9	8	7	6	5	4	3	2	1	
	(No entries)				F	C	O	P	E	R	
Tape feed				(button)							
First entry	0	0	0	0	0	0	(X)	X	X	X	Minimum register address
Program functions	0	0	0	0	X	X	(X)	X	X	X	Function code and operand
Last	0	0	0	0	0	0	0	0	0	0	End of Program
Tape feed				(button)							

All entries are made with the plus (+) bar.

(X) Entry in this column is legal for Quikomp II only.

Guide paper

Add-Punch Codes



OPERATING PROCEDURES

QUIKOMP SWITCH 3

Except for very simple Quikomp programs, it is wise to write a guide to operating procedures in advance. This can best be done by a review of the program sheets. Some points which should be included in the operating procedures are:

1. The starting register address.

2. List the sequence of input data and the ending function for each (tab or carriage return).

3. Use of the sense switches if required.

NUMERIC INPUT FROM TYPEWRITER - QUIKOMP I

To enter numeric data type the digits as normally written using a decimal point if required. Negative numbers should be preceded by a minus sign. The end of numeric input is indicated to the computer by use of the tab key (T) or the carriage return key (C/R).

EXAMPLES OF CORRECT INPUT:

1.5	(T) or (C/R)
.00593	(T) or (C/R)
-256	(T) or (C/R)

If a typing error is made before the end of input, touch the backspace key, then press the START button. Re-enter the entire value.

NUMERIC INPUT FROM TYPEWRITER - QUIKOMP II

To enter numeric data type the digits of the mantissa first. The decimal point is assumed to be located to the right of the most significant digit and it can be typed in this position for clarity. A negative mantissa should be preceded by a minus sign. The end of the mantissa is indicated to the computer by use of the slash key (/). If there is no exponent the end of numeric input is indicated by use of the tab key (T) or the carriage return key (C/R).

If an exponent is required it must be entered after the slash which followed the mantissa. A negative exponent should be preceded by a minus sign. The end of numeric input is indicated to the computer by use of the tab or the carriage return key.

EXAMPLES OF CORRECT INPUT:

1.5	(T) or (C/R)
5.93/-3	(T) or (C/R)
-2.56/2	(T) or (C/R)

If a typing error is made before the end of input, touch the backspace key, then press the START button. Re-enter the entire value.

ERROR CONTROL

Error controls are included in the Quikomp Program System to identify incorrect procedures to the programmer. The cause of the error and how to correct it is explained in this section.

Printed error messages are obtained when preparing a Quikomp program. Error codes are displayed in the control register lights when operating a Quikomp program.

PRINTED ERROR INDICATORS

SWITCH 1 AND 2 - LEARN AND CORRECT

PRINTOUT	CAUSE AND CORRECTION
FC ERROR	Illegal function code or code not ended by the tab key. Re-enter correct code followed by tab.
OPER ERROR	Illegal operand. Maximum register address is 320 (1,023 for Quikomp II). Re-enter correct operand.
E	Non-numeric key other than tab or carriage return used, or two keys touched together. Re-enter the function code or operand.
OVRFL ERR	Maximum register address has been exceeded. Re-write program to shorten it.

OPERATOR ERROR CONTROL

If an incorrect entry is made before touching the tab or carriage return key, type an "X". The computer will print an "E". Re-enter the function code or the operand.

DISPLAYED ERROR CODES

SWITCH 3 - EXECUTE

During operation of a Quikomp program errors will be indicated by codes appearing in control register lights numbered 8, 7, 6, 5, and 4, 3, 2, 1. The number 9 control register light and the RESET button will be lighted.

FOR QUIKOMP I AND II:

DISPLAY *	CAUSE AND CORRECTION
8765 4321	
0111 0001	Scaling overflow error, any function. Program will not continue.
0111 0011	Zero divisor. Program will not continue.
0111 0100	Square root of a negative number. Program will not continue.
0111 0101	Log of zero or negative number. Program will not continue.
0111 0110	Incorrect scaling, antilog. Program will not continue.
0111 1001	Typewriter input error. Press START, touch backspace, press START again, re-enter the entire number. - or - Tape reader input error. Raise tape gate, move tape back two characters, close gate and press START.
0111 1010	Typewriter input error key used. Press START and re-enter the entire number.

FOR QUIKOMP II ONLY:

0111 1000	Scaling overflow error, sine, cosine, tangent or cotangent. Program will not continue.
0111 1011	Input overflow error, more than 7 digits in mantissa or more than 2 digits in exponent. From typewriter, press START and re-enter the entire number. From tape, make corrected tape.

* 0 = not lighted; 1 = lighted.

OPERATOR ERROR CONTROL

If an incorrect entry is made from the typewriter before using the ending code, touch the backspace key. Press START and re-enter the entire number.

DISPLAYED ERROR CODES

SWITCH 4 - DATA PREPARATION

DISPLAY	CAUSE AND CORRECTION
8765 4321	
0111 1001	Typewriter input error. Press START, touch backspace, press START again, re-enter the entire number.
0111 1010	Typewriter input error key used. Press START and re-enter the entire number.

OPERATOR ERROR CONTROL

After typed data has been printed for verification, touch the space bar if correct to initiate punching. Touch the backspace key if incorrect, then type the correct data.

PRINTED ERROR INDICATORS

SWITCH 5 - PUNCH PROGRAM TAPE

PRINTOUT	CAUSE AND CORRECTION
TAPE IS OK	Program tape has been verified correct.
ERROR. MAKE A NEW TAPE	Program tape is incorrect, discard it and make a new tape.

SWITCH 6 - PROGRAM INPUT FROM ADD-PUNCH TAPE

PRINTOUT	CAUSE AND CORRECTION
ERROR	Illegal function code or operand. The Quikomp System will substitute a "no operation" function for that register and program input will continue. Use switch 2 to correct when program input is completed.
OVRFL ERR	If printed at the beginning of program input the starting register address does not appear on the tape. See operating instructions for switch 6, punching errors.
END	If END is not printed at the end of program input the "end of program" function code does not appear on the tape. See operating instructions for switch 6, punching errors.

PROGRAM TAPE INPUT ERRORS

The Quikomp Program System tape and program tapes produced by Quikomp control switch 5 are input to the computer by pressing the RESET button, typing 37X03008, press LOAD and START. If input of a program tape produces results described below, review appendix III and repeat the "Preparation for Program Input."

1. Program tape fails to read in or stops during input. Repeat "preparation for program input."

2. Program tape reads into computer but fails to operated properly. Repeat "program input."

3. Program tape stops, device #1 and parity lights are on, with 0010 0011 0000 0001 in control register lights.

   a. Tape reading error. Make sure tape is threaded correctly in reader and that the tape is not in a blank or 'visible character' area. Place tape in reader correctly and repeat from RESET.

   b. Tape reading error. Check for an odd number of holes ahead of the reader pins. Open tape gate, move tape back two characters, close tape gate and press START.

4. Program tape stops, device #2 light on, with 0010 0100 0000 0001 in control register lights. Tape is snagged or too tight in tape supply pan.

ADVANCED PROGRAMMING INFORMATION

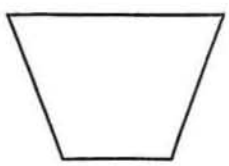
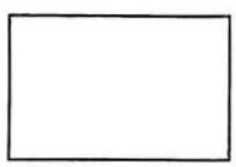
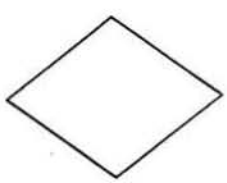
PLANNING A PROGRAM

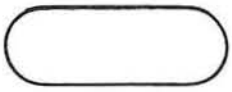
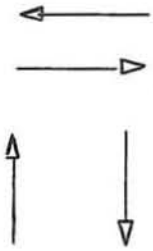
The "Procedure Description" for very simple programs can be written in English directly on the Quikomp Program Sheet. This is successful if the problem does not require more than two or three program sheets. More elaborate programs require very detailed planning.

Programmers use a set of symbols to chart the flow of program functions. Lines with arrowheads show the direction of movement from one function to the next. Each flowchart symbol contains a few English words and/or mathematical symbols to identify what is to take place in the program.

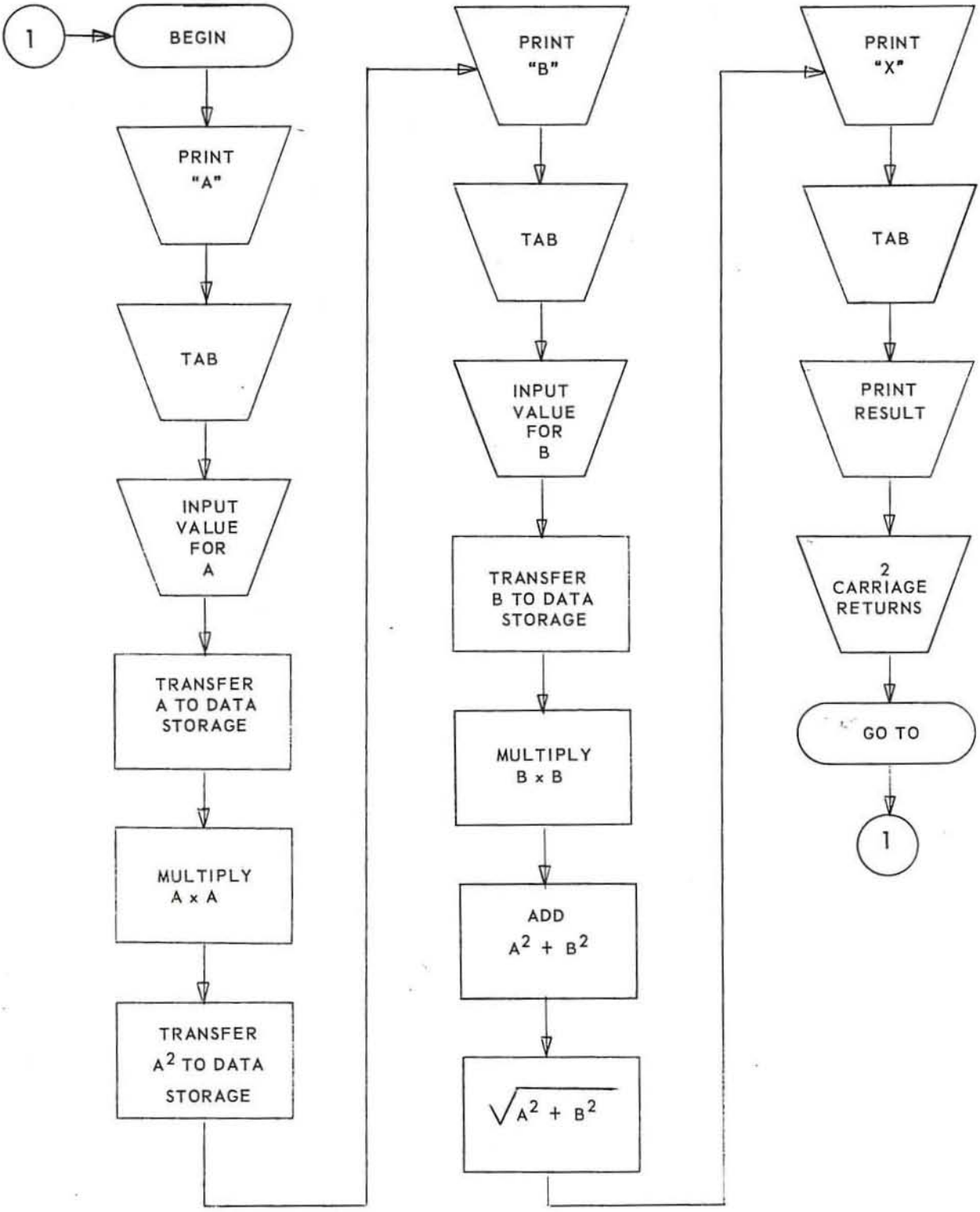
In general, a properly prepared flowchart should have a separate symbol for each Quikomp function. When the flowchart has been completed the symbols are translated to lines of procedure description on the Quikomp Program Sheet.

The following program flowchart symbols are used in the Quikomp Program System.

SYMBOL	DESCRIPTION
	Input or Output symbol. It is used for all Quikomp functions involving the typewriter, tape reader, or tape punch.
	Processing symbol. It is used for all Quikomp data transfer, arithmetic and mathematical functions. Also for no operation, pause, set loop counter, modify next operand, and sense switch functions.
	Decision symbol. It stands for a 'yes - no' decision or for testing a data value for positive, zero or negative. It is used for the Quikomp if negative, if zero, and loop return functions.

SYMBOL	DESCRIPTION
	Terminal symbol. This symbol is used for the beginning and ending points of a program. It is used with a connector as an interruption symbol for the Quikomp subroutine entry, subroutine exit, and go to functions.
	Connector symbol. This symbol stands for an exit to another part of the flowchart or for an entry from another part of the flowchart. A decimal number is written in both exit and entry connectors to link them together.
	Offpage connector. This symbol is used to indicate the page on which the flowchart is continued. The page number is written in the symbol.
	Direction symbols. A connector line with an arrowhead indicates the direction of movement on the flowchart. All flowchart symbols must be linked with connector lines except across a page.

EXAMPLE FLOWCHART FOR $\sqrt{A^2 + B^2}$



DATA VARIABLES AND PROGRAM CONSTANTS

Data variables are numeric values which will be different each time the program is operated. These values are called 'input data.' Input data can enter the computer from either the typewriter or the tape reader using a previously prepared data tape.

Program constants are numeric values which are required for calculations each time the program is operated. A constant is a number which does not change or vary. A constant is a necessary part of the program procedure.

Constants can be included in the Quikomp program system by one of two methods. If only a few constants are required they should be entered from the typewriter at the beginning of the program. The program is repeated from the point where the first variable data is entered. (See Method 1 example.)

If a group of constants is required in a program they should be entered from the typewriter or tape reader using the three looping functions. (See method 2 example.) This procedure is

1. Enter the six step program used to input a group of constants. The program may occupy any register addresses except those where the constants will be placed.

2. Run the program to input the constants and place them in the selected register addresses. This program will be used only once; after the constants have been input it will not be required again.

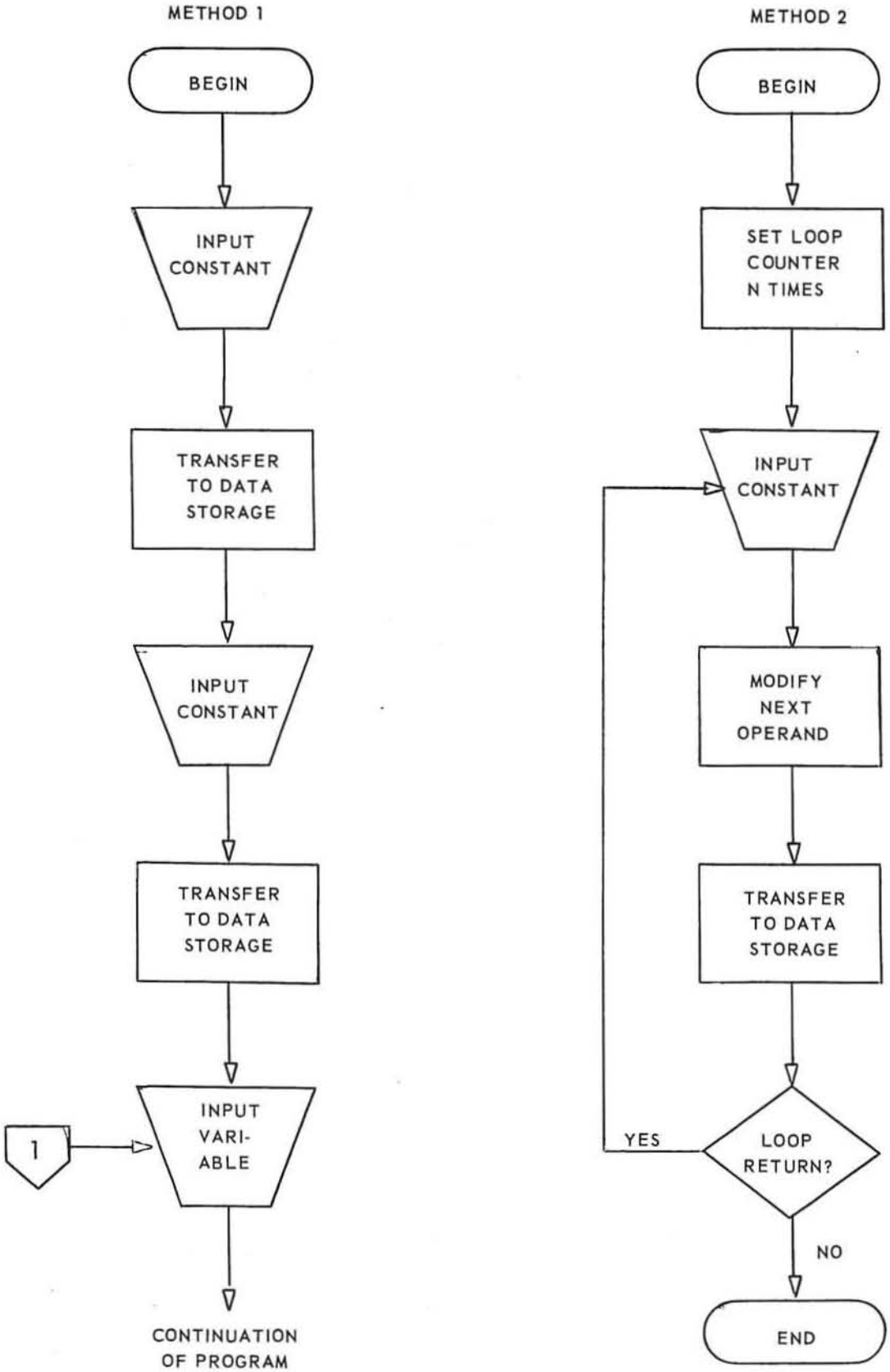
3. Enter the program which requires the constants already input to the computer.

4. Test the program with pre-calculated data.

5. When all portions of the program have been tested, punch the program procedure registers <u>and</u> the registers containing the program constants into a single program tape using Quikomp switch 5.

6. The program constants have been captured as part of the program procedure. Both are a part of the program tape which will be read into the computer when the program is to be used again.

INPUT OF PROGRAM CONSTANTS



USE OF STORAGE REGISTERS

A computer program contains procedure functions and numeric constants and requires data variables. Computer storage registers must be set aside for each of these three items. Careful, systematic planning will result in the most efficient use of computer storage.

Rule 1: Contants should be placed in storage register addresses beginning with register 1. The number of constants required is always known for any program.

Rule 2: Program procedure functions should follow immediately after the registers containing constants. No unused registers should be 'reserved.' This permits the procedure and the constants to be output to a single program tape.

Rule 3: Storage registers for input data variables should use the highest numbered register addresses. This allows for addition or expansion of the program without major changes. Data storage registers are for temporary use and are not part of the program procedure. These locations do not have to be output to a program tape for future use.

Rule 4: Use a Quikomp Program Sheet to indicate the use of register addresses which contain constants or variable data. The value of the constant or an identification of the data should be written on the 'Procedure Description' line. Nothing is entered in the Code or Operand columns.

APPENDIX I

BASIC COMPUTER OPERATION

EQUIPMENT REQUIREMENTS

The Quikomp Programming System requires a Monrobot XI computer with either 1,024 or 2,048 registers of storage. An input/output typewriter (either a 16" or 20" carriage) is connected to input #1. (See "device Connections" in Appendix II). A punched paper tape reader is connected to input #2, and a paper tape punch to output #2.

DESCRIPTION OF CONSOLE FUNCTIONS

POWER ON BUTTON

This button, located in the lower left corner of the control panel, sends power to the computer and its devices. Pressing the button will light it and turn on power to the computer. After the computer is turned on approximately one minute is required for the storage drum to attain maximum speed and the electronic portion to warm up. The computer is ready to operate when the RESET button lights.

When the computer is ON, pressing the power button will turn the computer OFF. If power is turned off or fails during operation of the computer the program must be re-started. The program is not lost from computer storage, only the data being operated on at the moment of power failure will be lost.

RESET BUTTON

The computer is in the proper mode for entry of a starting instruction when the RESET button is lighted. Pressing the RESET button during operation of a program will cause the computer program to cease operation. The program must then be re-started from a known entry point. The RESET button will light when the PAUSE instruction is executed. Do not press the RESET button at that time unless a new starting instruction is to be entered.

Each time the RESET button lights the #9 neon lamp in the control register display will also be lighted.

LOAD BUTTON

To begin operation of a computer program, eight characters must be entered from the typewriter keyboard when the RESET light is on. This is called reset entry. Legal entry characters are the digits 1 through 9, zero, and the alphabetic characters S through X. Note that there are separate keys for the digits 1 and zero; do not use the alphabetic characters L or O. The carriage return function may be used prior to reset entry but not following.

Pressing the LOAD button transfers the entry instruction to the Control Register of the computer where it is processed when the START button is pressed. Pressing the LOAD button during computer operation will have no effect.

START BUTTON

After entry of a starting instruction and use of the LOAD button, pressing the START button will cause the computer to execute the starting instruction. Pressing the START button during operation of a computer program will have no effect. The START button must be used to cause computer operation to resume following a PAUSE instruction. The START button is always lighted during operation of a computer program.

HALT BUTTON

Pressing the HALT button during operation of a program will cause the computer to stop. When the HALT button is pressed it will light and the START button light will go off. To resume operation, press the HALT button a second time (the light will go off), then press the START button and the computer program will continue. The HALT button will not be lighted for a PAUSE instruction.

SENSE SWITCHES

The switches are numbered 1 through 8 and are located in the middle of the control panel above the ON, RESET, LOAD, HALT and START buttons. When pressed the first time the switch will light and is considered to be ON. When pressed again the light will be out and the switch is OFF. The switches are normally OFF unless program operating instructions state otherwise.

INPUT/OUTPUT DEVICE LIGHTS

The upper row of round lights above the sense switches will indicate when input is expected from a device or output is intended to a device. From left to right these lights indicate the following:

Top left: Input from the typewriter is expected (device #1).

Second from left: Input from the tape reader is expected (device #2).

Third and fourth from left: Not used in the Quikomp System.

(Parity light, see following)

Third from right: Output to the typewriter is intended (device #1).

Second from right: Output to the tape punch is intended (device #2).

Top right: Not used in the Quikomp System.

PARITY LIGHT

This light is located between the input and output lights and is labeled. Its purpose is to indicate a mis-read or incorrect character from an input device. In the Quikomp system this type of error is normally displayed in the control register lights. The single exception to this rule is described in Appendix III, paragraph 4.

CONTROL REGISTER LIGHTS

The 16 neon lights at the bottom of the control panel display each computer instruction before it is performed. There are 4 groups of 4 lights numbered 1 to 16 from right to left. The Quikomp PAUSE function code and all error codes are displayed in the control register lights. The explanation of these displays is given in this manual under "Error Codes."

The four groups of neon lamps will have the following appearance:

CONTROL REGISTER DISPLAY LIGHTS

0	0	0	0	0	0	0	1	C	C	C	C	C	C	C	C
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

(Numbers at bottom of control panel)

When an error code is displayed in the control register lights the reset button will be lighted, the start button will be off. The lights numbered 16 through 10 will be off, light number 9 will always be on. The two groups of four lights each, 8, 7, 6, 5 and 4, 3, 2, 1, are assigned a code symbol according to the table below.

CCCC CODE	SYMBOL
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	S
1011	T
1100	U
1101	V
1110	W
1111	X

A C code of 0 means not lighted; a C code of 1 means lighted.

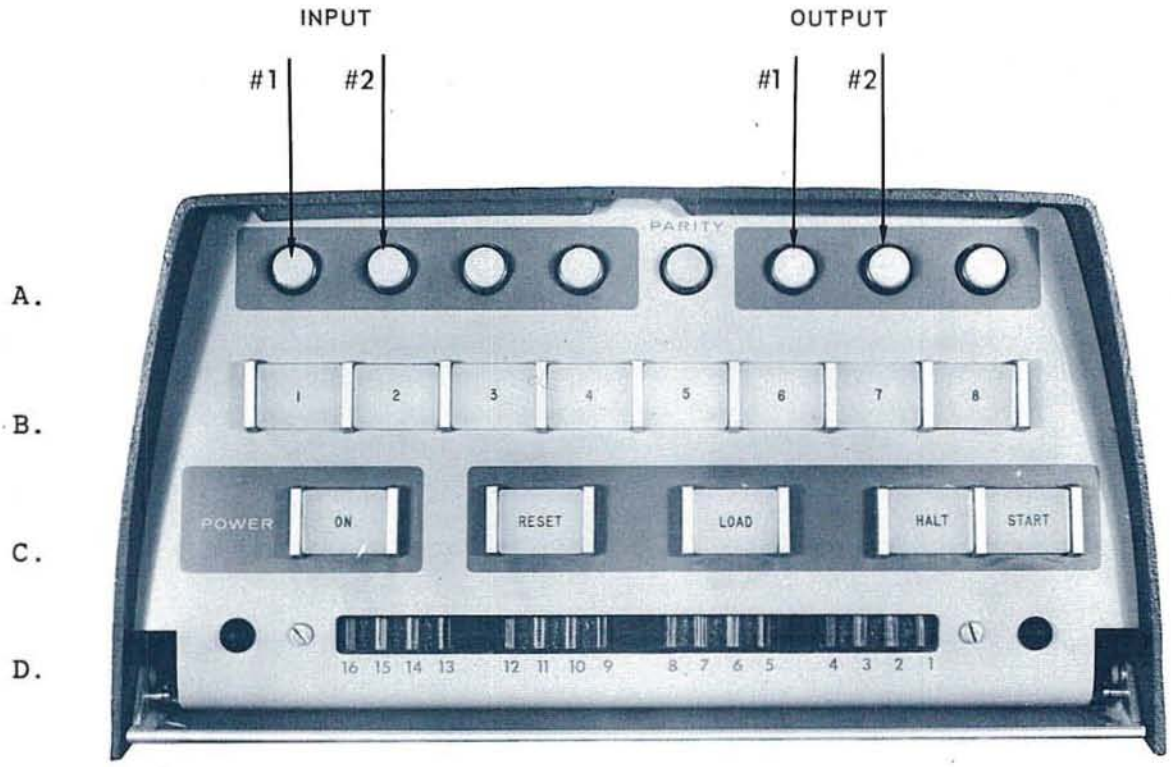
CARE OF PUNCHED PAPER TAPES

Each tape produced by the Quikomp Programming System should be identified. Using a pencil write the name of the program and the date on the end of the tape.

Tapes which are used frequently are subject to wear and should be duplicated from time to time. The procedure for doing this is found in Appendix II, Tape Reader and Punch, page 7. A duplicate copy of the Quikomp System tapes should be made and placed aside for safekeeping. This is insurance against accidentally torn or worn program tapes.

Punched paper tape should be stored in plastic or metal boxes or drawers. Avoid crushing or folding of tapes and keep them free from dirt. Paper clips, if used, should be applied and removed with care.

CONTROL PANEL LAYOUT



Row A. Input/Output and parity lights.

Row B. Sense switches.

Row C. Power On/Off and control buttons.

Row D. Control register display lights.

APPENDIX II - DEVICE OPERATION

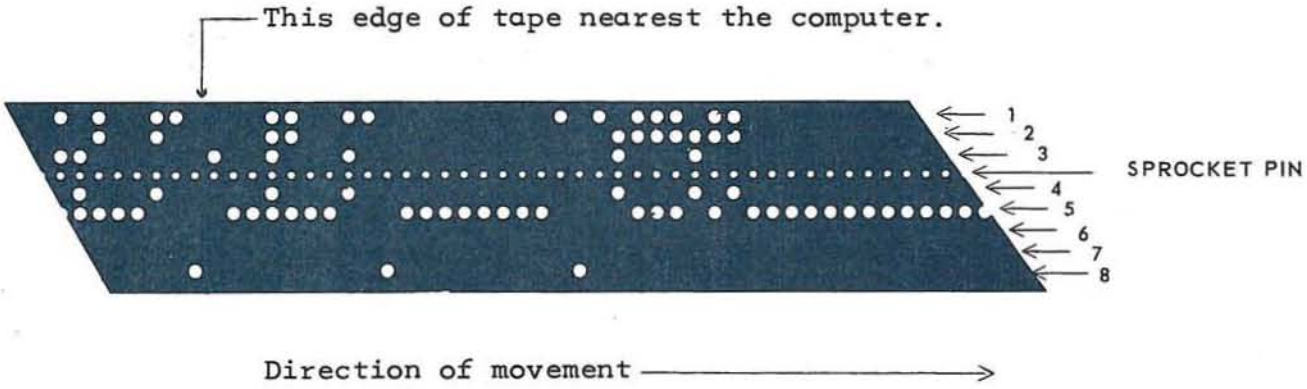
1. Input/Output Typewriter
2. Tape Reader and Punch
3. Device Connections to the Computer

APPENDIX III

PREPARATION FOR PROGRAM INPUT

The computer must always contain a group of instructions which cause a program tape to be input. The procedure described below must be repeated any time the computer fails to input a program tape correctly.

A special tape, Quikomp PRGM LOADER, is provided to record the program input instructions in computer storage registers. Place this tape in the reader with the pins in the single punch in the fifth channel.



Press the RESET button.

Type T3W1S380	press LOAD,	Type 240173W5	press START.	A carriage return will occur each time START is pressed.
" T3W2S380	"	U0059008	"	
" T3W3S380	"	X005T3W6	"	
" T3W4S380	"	T00533W1	"	
" T3W5S380	"	2400X3W3	"	

Type 33W10000 press LOAD and START buttons.

As the tape is read into the computer the "input, device #2" and "parity" lights will flash when every ninth character is read (the single punch in the eighth channel). Reading will stop at the end of the tape.

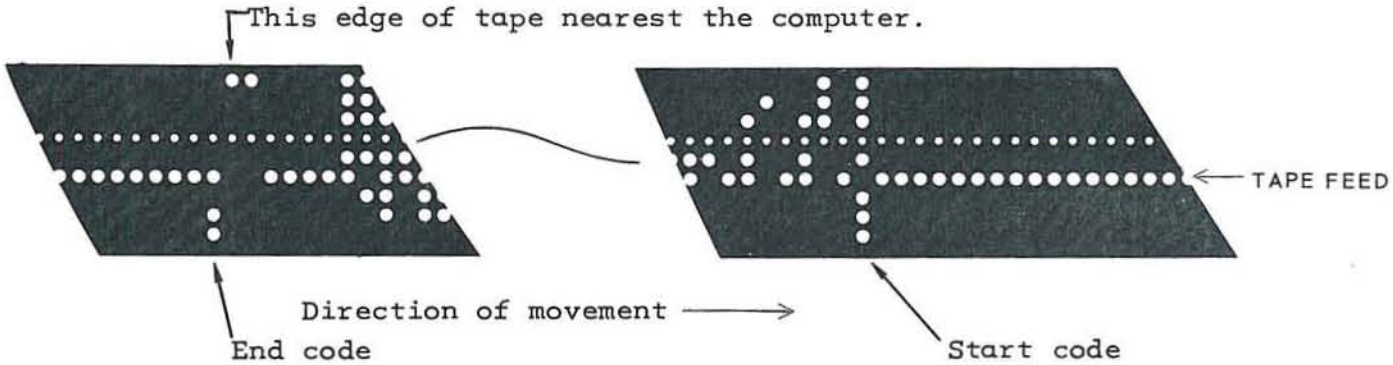
INPUT OF A PROGRAM TAPE

Specific procedures are given in this manual for input of programs in the Quikomp System. In general the sequence of operations is:

1. Press the RESET button on the front of the control panel.

2. Place the program tape in the reader supply pan so that it unwinds clockwise from the center. Open the tape reader gate and thread the tape behind the wire guide, then across the top of the tape reader under the two guides and the reader comb. Make certain that the smallest holes are correctly located on the rotating sprocket drive pins. Close the tape reader gate.

The tape should look like this:



3. Type 37X03008 and press the LOAD and START buttons. The program tape will be input to the computer and scan the Quikomp control switches.

4. If a reading error occurs the input, device #1 light, and the parity light will come on and input will stop. Open the tape reader gate, space the tape back two characters using the knob on the front of the reader, close the gate and touch the space bar on the typewriter. Input will continue.

APPENDIX IV

The example programs in this section were selected to illustrate Quikomp program functions rather than the computational ability of the system. The examples include common programming techniques such as loops, logical decisions, and the use of sense switches.

PROBLEM 1

ANALYSIS:

Square root of A^2 plus B^2 $\quad X = \sqrt{A^2 + B^2}$

Calculate X for any values of A and B where A and B will be input from the typewriter.

PRINTED FORMAT

A	(tab)	Enter value
B	(tab)	Enter value
X	(tab)	Print result

The flowchart for problem 1 is on page 61 of this manual.

SAMPLE DATA AND RESULTS

```
A        3  <------ ENTER FROM TYPEWRITER,
B        4           END WITH CARRIAGE RETURN.
X         5.000000

A        1
B        1
X         1.414214

A        45
B        66
X         79.88116

A        34.678
B        22.99986
X         41.61199

A        1.2
B        2.4
X         2.683282

A        .00567
B        .02345
X         .02412574
```

MONROBOT XI QUIKOMP™ PROGRAM SHEET		PAGE 1 OF 1
PROGRAM $X = \sqrt{A^2 + B^2}$	PROGRAMMER G. S.	DATE / /

PROCEDURE DESCRIPTION	REGISTER	CODE	OPERAND
Print letter A	1	2	49
Tab typewriter	2	13	
Input value for A	3	1	
Transfer A to storage	4	10	23
Multiply A × A	5	32	23
Transfer A^2 to storage	6	10	24
Print letter B	7	2	50
Tab typewriter	8	13	
Input value for B	9	1	
Transfer B to storage	10	10	25
Multiply B × B	11	32	25
Add $A^2 + B^2$	12	28	24
Square root of $A^2 + B^2$	13	27	
Print letter X	14	2	87
Tab typewriter	15	13	
Print result	16	5	
Carriage return	17	11	
Carriage return	18	11	
Go to 1 to repeat	19	12	1
End of program	20	0	
	1		
Data Storage	2		
Storage area for A	23		
" " " A^2	24		
" " " B	25		
	6		
	7		
	8		
	9		
	0		

PROBLEM 2

ANALYSIS

FACTORIAL EQUATION N!

Calculate N! for any value of N where N will be input from the typewriter. N is defined as a positive integer.

The formula to calculate N! is:

$$N(N-1)\ (N-2)\ (N-3)\ldots\ldots\ldots 1$$

Problem: Find the largest N for which N! can be obtained on the computer.

PRINTED FORMAT

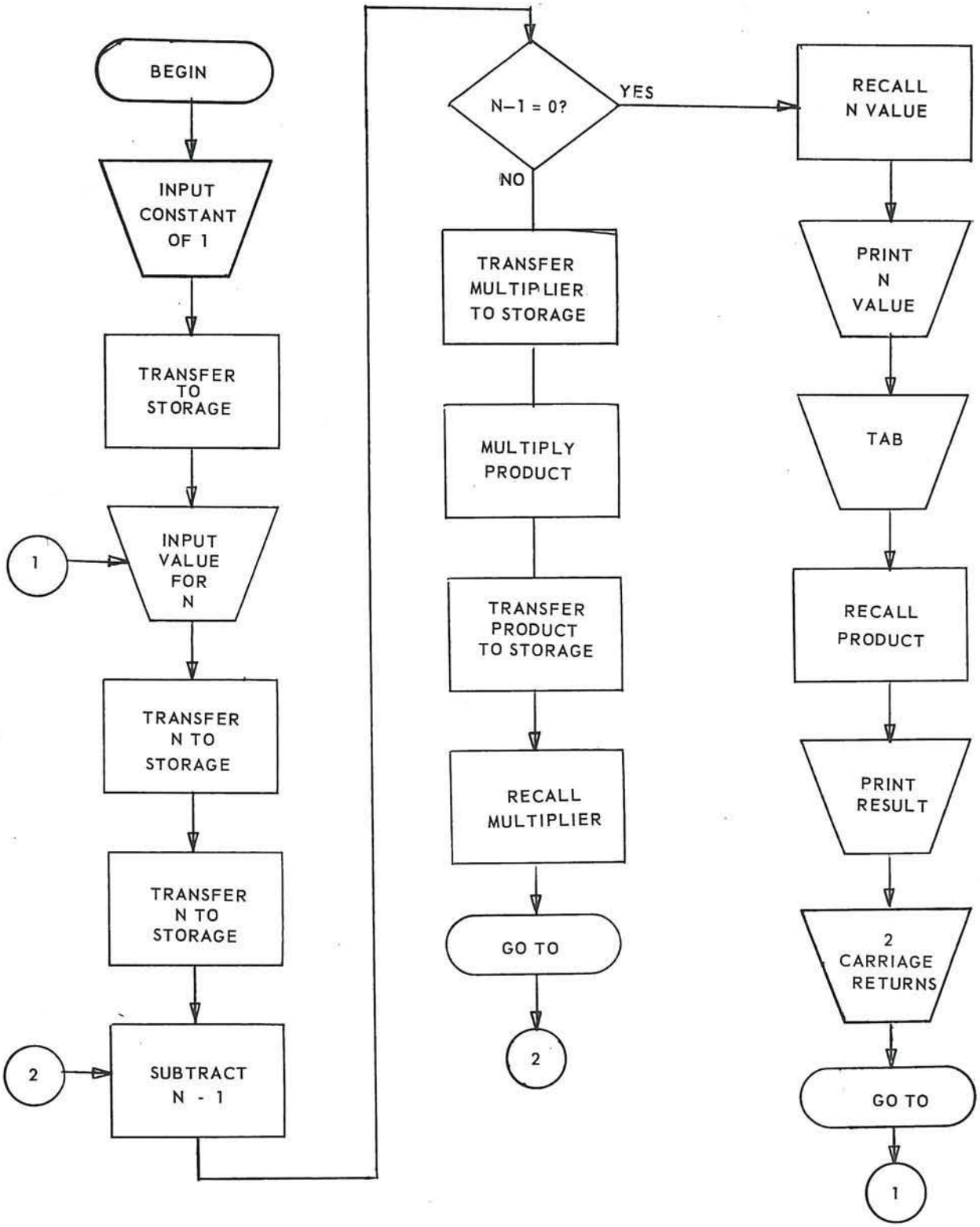
```
1       (Enter constant of 1 to be used as subtractor.  Required once for
         the first operation of the program.)
N       (Enter value for N)
N value prints                  (tab)              Result

N       (Enter value for N)
etc.
```

SAMPLE DATA AND RESULTS

1 ←	ENTER FROM TYPEWRITER, END WITH CARRIAGE RETURN.
3	
3.000000	6.000000
4	
4.000000	24.00000
5	
5.000000	120.0000
9	
9.000000	362880.0
7	
7.000000	5040.000

FACTORIAL EQUATION



MONROBOT XI QUIKOMP™ PROGRAM SHEET		PAGE 1 OF 1
PROGRAM N!	PROGRAMMER T. N.	DATE / /

PROCEDURE DESCRIPTION	REGISTER	CODE	OPERAND
Input constant 1	1	1	
Transfer 1 to storage	2	10	27
Input N value	3	1	
Transfer N to storage	4	10	28
Transfer N to storage	5	10	29
Subtract (N - 1)	6	30	27
Test N-1 for 0	7	16	13
Transfer multiplier to storage	8	10	30
Multiply product	9	32	29
Transfer product to storage	10	10	29
Recall multiplier	11	8	30
Go to 6 to repeat multiplication	12	12	6
Recall N value	13	8	28
Print N value	14	5	
Tab typewriter	15	13	
Recall product	16	8	29
Print result	17	5	
Carriage return	18	11	
Carriage return	19	11	
Go to 3 to repeat program	20	12	3
End of program	21	0	
	2		
	3		
	4		
	5		
Data Storage	6		
Storage area for 1	27		
" " " N	28		
" " " product	29		
" " " multiplier	30		

PROBLEM 3

ANALYSIS:

COMPOUND INTEREST $A = P(1 + r/m)^{mn}$

Calculate the accumulated amount, A, of a principal, P, placed at a rate of interest r for n years compounded m times a year.

The procedure to be used in calculating the amount is as follows:

```
  log of (1+r/m) times mn
+ log of P
  log of A = Sum of mnlog (1+r/m) + log P
         A = Antilog of log A
```

The values 1,r,m,n, and P will be input from the typewriter. If switch 8 is on, m, n, and r will remain constant for each calculation but the value of P will be variable.

PRINTED FORMAT

```
K   (tab)   Enter constant of 1
R   (tab)   Enter r value
M   (tab)   Enter m value
N   (tab)   Enter n value
P   (tab)   Enter P value
A   (tab)   Result prints

P   (tab)   Enter P value   Switch 8 on
A   (tab)   Result prints

etc.
```

NOTE: The constant of 1 is entered once for the first operation of the program.

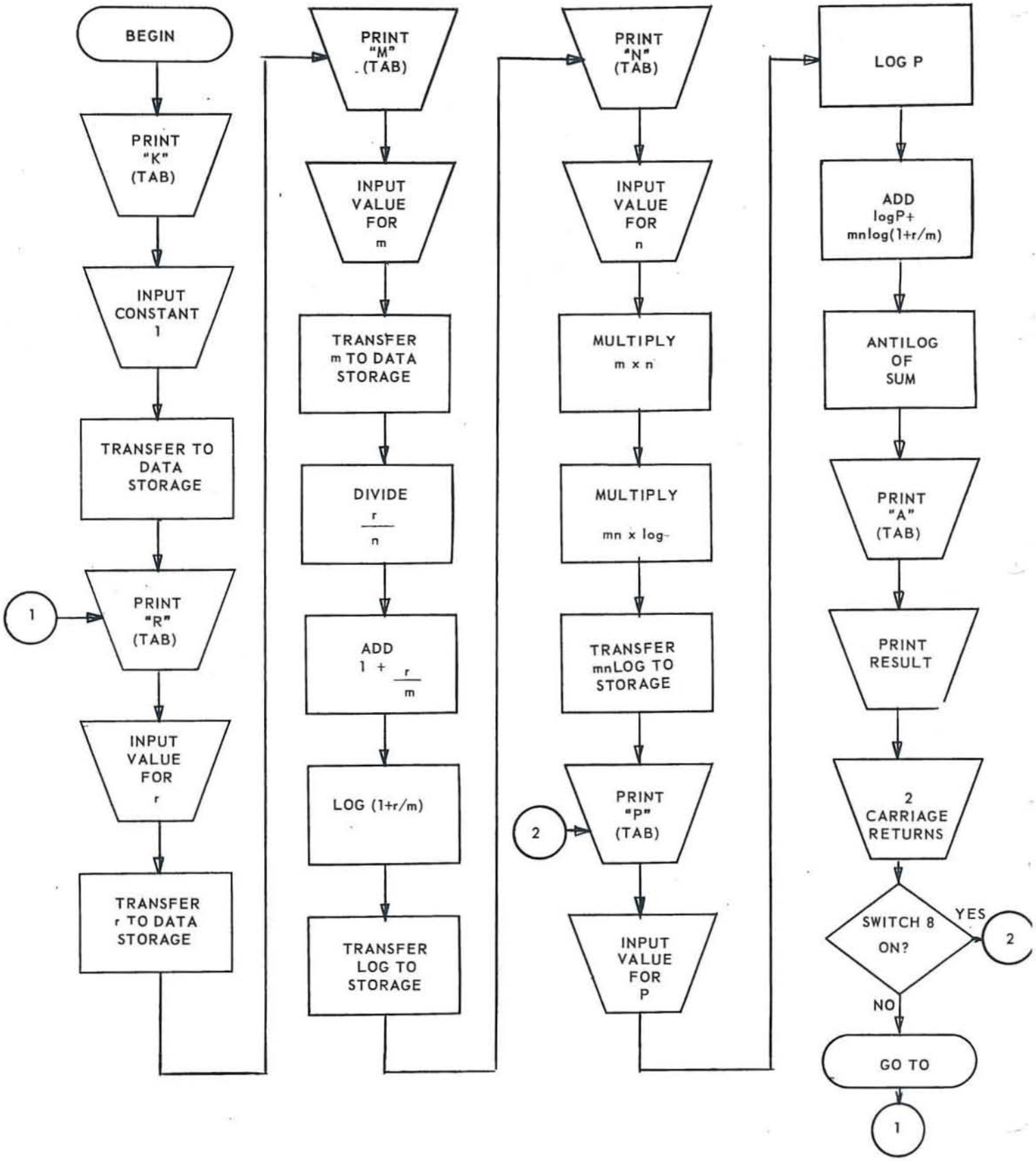
SAMPLE DATA AND RESULTS

```
K                1
R                .08
M                2
N                20
P                4000
A                 19203.70
(SWITCH 8 OFF)
R                .045
M                2
N                25
P                6000
A                 18252.06
(SWITCH 8 ON)
P                9000
A                 27378.14

P                10000
A                 30420.12
```

ENTER FROM TYPEWRITER, END WITH CARRIAGE RETURN.

COMPOUND INTEREST



MONROBOT XI QUIKOMP™ PROGRAM SHEET		PAGE 1 OF 2
PROGRAM $A = P(1 + \frac{r}{m})^{mn}$	PROGRAMMER T.H.	DATE / /

PROCEDURE DESCRIPTION	REGISTER	CODE	OPERAND
Print letter K	1	2	98
Tab typewriter	2	13	
Read typewriter (1)	3	1	
Transfer 1 to storage	4	10	40
Print letter R	5	2	41
Tab typewriter	6	13	
Read typewriter (r)	7	1	
Transfer r to storage	8	10	41
Print letter M	9	2	100
Tab typewriter	10	13	
Read typewriter (m)	11	1	
Transfer m to storage	12	10	42
Divide r/m	13	34	41
Add 1 + r/m	14	28	40
Log (1 + r/m)	15	31	
Transfer log (1+r/m) to storage	16	10	43
Print letter N	17	2	37
Tab typewriter	18	13	
Read typewriter (n)	19	1	
Multiply m × n	20	32	42
Multiply mn × log (1+r/m)	21	32	43
Transfer mn log (1+ r/m) to storage	22	10	44
Print letter P	23	2	103
Tab typewriter	24	13	
Read typewriter (P)	25	1	
Log P	26	31	
Add log P + mn log (1+r/m)	27	28	44
Antilog of log P + mn log (1+r/m)	28	35	
Print letter A	29	2	49
Tab typewriter	30	13	

MONROBOT XI QUIKOMP™ PROGRAM SHEET		PAGE 2 OF 2
PROGRAM $A = P(1 + \frac{r}{m})^{mn}$	PROGRAMMER T. M.	DATE / /

PROCEDURE DESCRIPTION	REGISTER	CODE	OPERAND
Write result	31	5	
Carriage return	32	11	
Carriage return	33	11	
Test switch 8 for on	34	20	128
If negative, switch 8 on	35	14	23
Go to 5 to repeat	36	12	5
End of program	37	0	
	8		
Data Storage	9		
Storage area for constant 1	40		
" " " r	41		
" " " m	42		
" " " log (1 + r/m)	43		
" " " mn log (1 + r/m)	44		
	5		
	6		
	7		
	8		
	9		
	0		
	1		
	2		
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	0		

PROBLEM 4

ANALYSIS:

QUADRATIC EQUATION $AX^2 + BX + C = 0$

Calculate X for any values of A, B, and C where A, B, and C will be input from the typewriter.

The following formula will be used:

$$X = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

If A = 0, the equation is linear and X = -C/B

If $B^2 - 4AC > 0$, the roots are real and unequal

If $B^2 - 4AC = 0$, the roots are real and equal

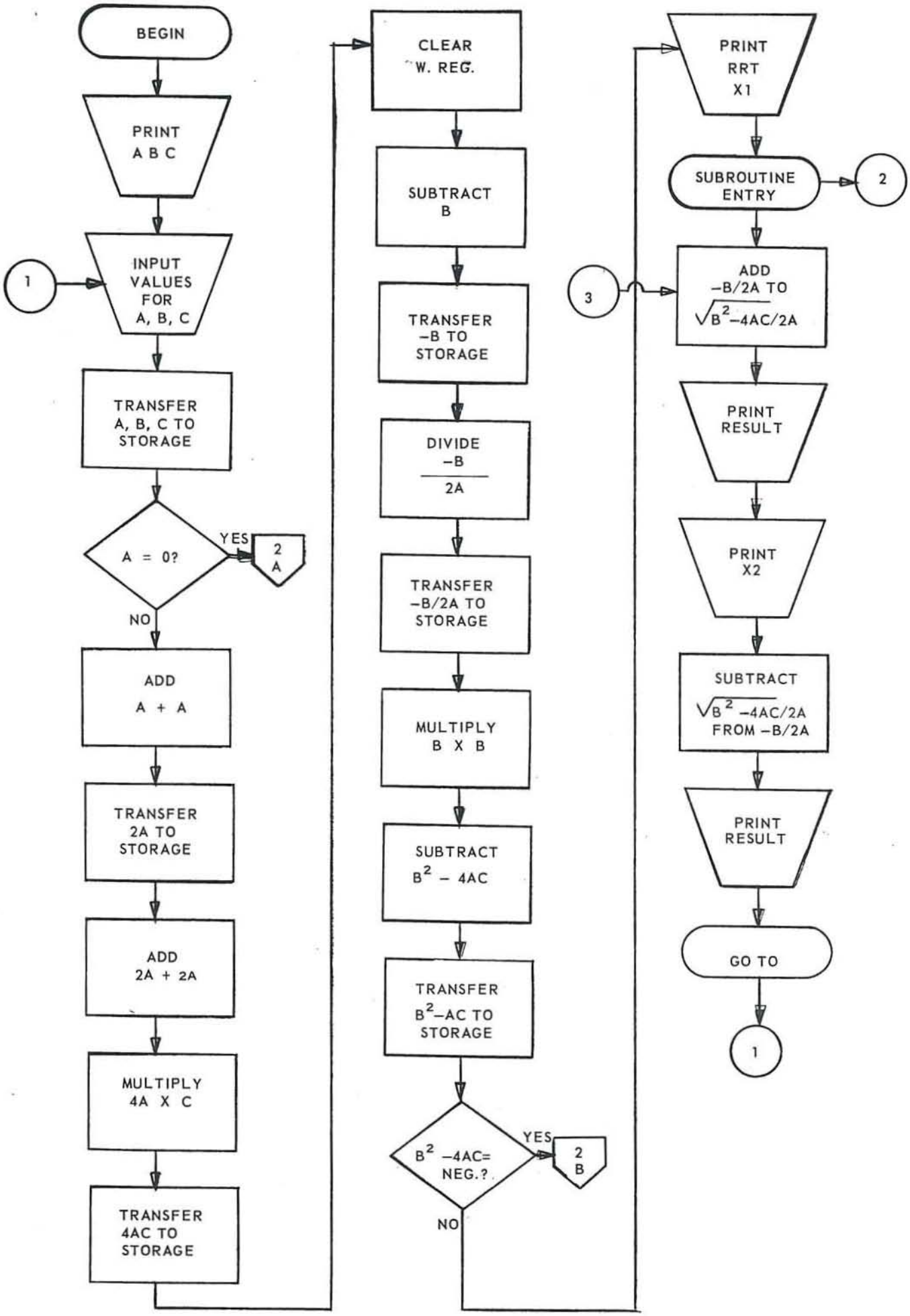
If $B^2 - 4AC < 0$, the roots are imaginary

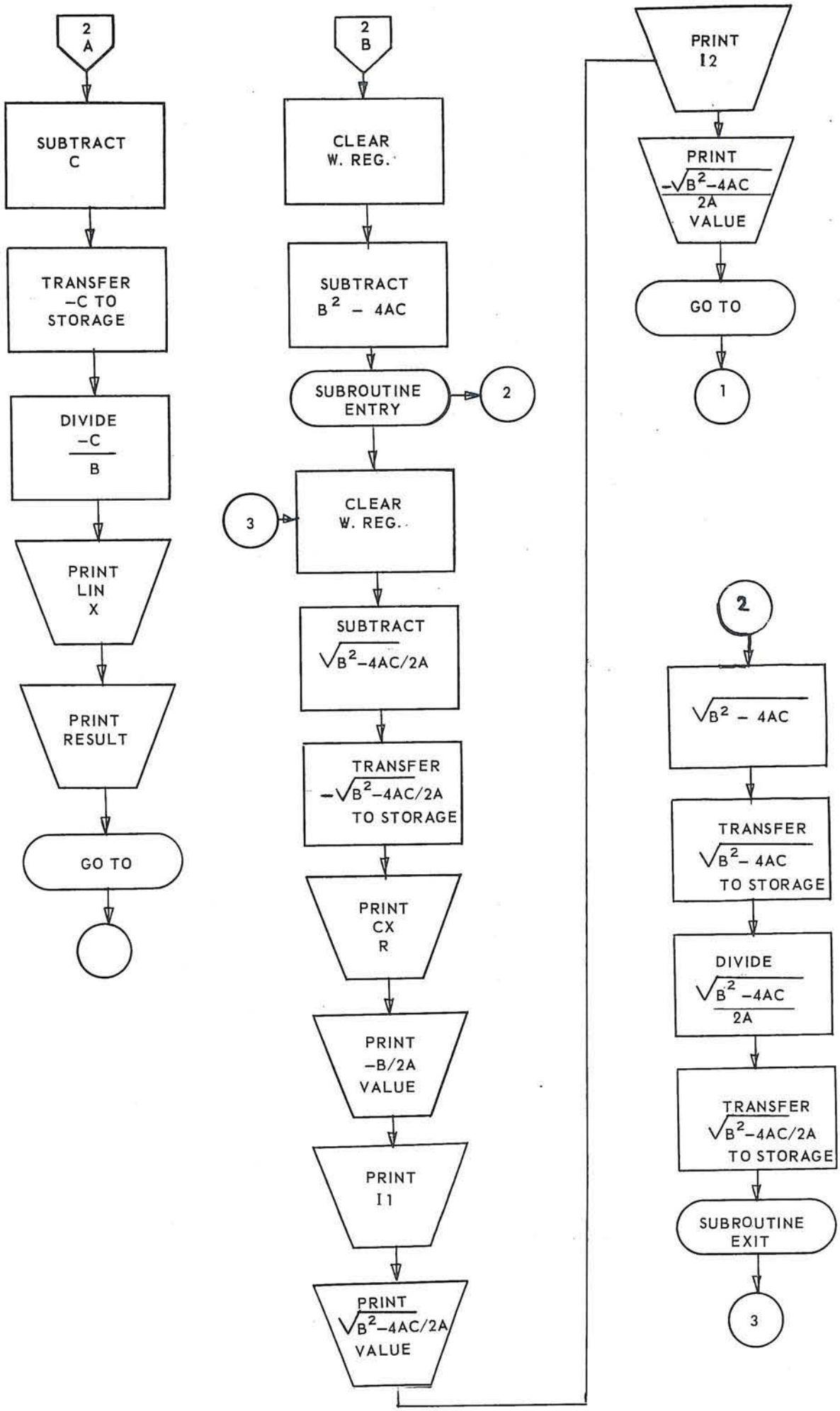
PRINTED FORMAT

	A	B	C
(Linear)	0 LIN X Result	B value	C value
(Real Roots)	A value R RT X1 Result X2 Result	B value	C value
(Complex roots)	A value CX R Real portion of result I1 Imaginary portion of result I2 Imaginary portion of result	B value	C value

SAMPLE DATA AND RESULTS

A	B	C
0 (END ON TAB)	2 (END ON TAB)	6 (END ON C/R)
LIN		
X -3.000000		
2	5	2
R RT		
X1 -.5000000		
X2 -2.000000		
1	1	1
CX		
R -.5000000		
I1 .8660255		
I2 -.8660255		
3	5	9
CX		
R -.8333333		
I1 1.518406		
I2 -1.518406		
15	43	67
CX		
R -1.433333		
I1 1.553133		
I2 -1.553133		
0	19	-4
LIN		
X .2105263		
2.55	14.78	4.1234
R RT		
X1 -.2938860		
X2 -5.502192		
53.6	98.2	2
R RT		
X1 -.02059820		
X2 -1.811491		


BEGIN
PRINT A B C
1
INPUT VALUES FOR A, B, C
TRANSFER A, B, C TO STORAGE
A = 0?
YES
2 A
NO
ADD A + A
TRANSFER 2A TO STORAGE
ADD 2A + 2A
MULTIPLY 4A X C
TRANSFER 4AC TO STORAGE
CLEAR W. REG.
SUBTRACT B
TRANSFER −B TO STORAGE
DIVIDE −B / 2A
TRANSFER −B/2A TO STORAGE
MULTIPLY B X B
SUBTRACT B² − 4AC
TRANSFER B²−AC TO STORAGE
B² −4AC= NEG.?
YES
2 B
NO
PRINT RRT X1
SUBROUTINE ENTRY
2
3
ADD −B/2A TO √B²−4AC/2A
PRINT RESULT
PRINT X2
SUBTRACT √B² −4AC/2A FROM −B/2A
PRINT RESULT
GO TO
1


2 A
SUBTRACT C
TRANSFER −C TO STORAGE
DIVIDE −C / B
PRINT LIN X
PRINT RESULT
GO TO
2 B
CLEAR W. REG.
SUBTRACT B² − 4AC
SUBROUTINE ENTRY
2
3
CLEAR W. REG.
SUBTRACT √B²−4AC/2A
TRANSFER −√B²−4AC/2A TO STORAGE
PRINT CX R
PRINT −B/2A VALUE
PRINT I1
PRINT √B²−4AC/2A VALUE
PRINT I2
PRINT −√B²−4AC / 2A VALUE
GO TO
1
2
√B² − 4AC
TRANSFER √B²− 4AC TO STORAGE
DIVIDE √B² −4AC / 2A
TRANSFER √B²−4AC/2A TO STORAGE
SUBROUTINE EXIT
3

MONROBOT XI QUIKOMP™ PROGRAM SHEET		PAGE 1 OF 4
PROGRAM: Quadratic Equation	PROGRAMMER: M. I.	DATE: / /

PROCEDURE DESCRIPTION	REGISTER	CODE	OPERAND
Print letter A	1	2	49
Tab typewriter	2	13	
Print letter B	3	2	50
Tab typewriter	4	13	
Print letter C	5	2	115
Carriage return	6	11	
Carriage return	7	11	
Set loop counter to 3	8	24	3
Read typewriter	9	1	
Modify next operand	10	25	
Transfer to storage (A, B, C)	11	10	100
Loop return to read again	12	26	8
Recall A value	13	8	100
Test A for 0	14	16	50
Add (A + A)	15	28	100
Transfer 2A to storage	16	10	103
Add (2A + 2A)	17	28	103
Multiply (4A × C)	18	32	102
Transfer 4AC to storage	19	10	104
Clear working register	20	23	
Subtract (0 − B)	21	30	101
Transfer −B to storage	22	10	105
Recall 2A	23	8	103
Divide (−B/2A)	24	34	105
Transfer −B/2A to storage	25	10	107
Recall B	26	8	101
Multiply (B × B)	27	32	101
Subtract (B^2 − 4AC)	28	30	104
Transfer B^2 − 4AC to storage	29	10	106
Test B^2 − 4AC for negative	30	14	62

MONROBOT XI QUIKOMP™ PROGRAM SHEET		PAGE 2 OF 4
PROGRAM Quadratic Equation	PROGRAMMER M. I.	DATE / /

PROCEDURE DESCRIPTION	REGISTER	CODE	OPERAND
Print letter R	31	2	41
Print space	32	2	64
Print letter R	33	2	41
Print letter T	34	2	19
Carriage return	35	11	
Print letter X	36	2	87
Print number 1	37	2	1
Print space	38	2	64
Subroutine entry	39	18	88
Add $(\sqrt{B^2-4AC}/2A - B/2A)$	40	28	107
Write result	41	5	
Carriage return	42	11	
Print letter X	43	2	87
Print number 2	44	2	2
Print space	45	2	64
Recall $-B/2A$	46	8	107
Subtract $(-B/2A - \sqrt{B^2-4AC}/2A)$	47	30	109
Write result	48	5	
Go to repeat	49	12	6
Subtract C	50	30	102
Transfer -C to storage	51	10	110
Recall B	52	8	101
Divide $(-C/B)$	53	34	110
Print letter L	54	2	35
Print letter I	55	2	121
Print letter N	56	2	37
Carriage return	57	11	
Print letter X	58	2	87
Print space	59	2	64
Write result	60	5	

MONROBOT XI QUIKOMP™ PROGRAM SHEET		PAGE 3 OF 4
PROGRAM Quadratic Equation	PROGRAMMER M. I.	DATE / /

PROCEDURE DESCRIPTION	REGISTER	CODE	OPERAND
Go to 6 to repeat	61	12	6
Clear working register	62	23	
Subtract $(0 - (B^2 - 4AC))$ - Positive	63	30	106
Subroutine entry	64	18	88
Clear working register	65	23	
Subtract $(0 - (\sqrt{B^2-4AC}/2A))$	66	30	109
Transfer $-\sqrt{B^2-4AC}/2A$ to storage	67	10	111
Print letter C	68	2	115
Print letter X	69	2	87
Carriage return	70	11	
Print letter R	71	2	41
Print space	72	2	64
Recall $-B/2A$	73	8	107
Write result	74	5	
Carriage return	75	11	
Print letter I	76	2	121
Print letter 1	77	2	1
Print space	78	2	64
Recall $\sqrt{B^2-4AC}/2A$	79	8	109
Write result	80	5	
Carriage return	81	11	
Print letter I	82	2	121
Print number 2	83	2	2
Print space	84	2	64
Recall $-\sqrt{B^2-4AC}/2A$	85	8	111
Write result	86	5	
Go to 6 to repeat	87	12	6
Square root of $B^2 - 4AC$	88	27	
Transfer $\sqrt{B^2-4AC}$ to storage	89	10	108
Recall 2A	90	8	103

MONROBOT XI QUIKOMP™ PROGRAM SHEET

PROGRAM: Quadratic Equation

PROGRAMMER: M. I.

DATE: / /

PROCEDURE DESCRIPTION	REGISTER	CODE	OPERAND
Divide $\sqrt{B^2-4AC}/2A$	91	34	108
Transfer $\sqrt{B^2-4AC}/2A$ to storage	92	10	109
Subroutine exit	93	19	
End of program	94	0	
	5		
	6		
	7		
	8		
Data Storage	9		
Storage area for A	100		
" " " B	101		
" " " C	102		
" " " 2A	103		
" " " 4AC	104		
" " " $-B$	105		
" " " B^2-4AC	106		
" " " $-B/2A$	107		
" " " $\sqrt{B^2-4AC}$	108		
" " " $\sqrt{B^2-4AC}/2A$	109		
" " " $-C$	110		
" " " $-\sqrt{B^2-4AC}/2A$	111		
	2		
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	0		