THE BENDIX CORPORATION COMPUTER DIVISION
5630 Arbor Vitae Street, Los Angeles 45, California

TECHNICAL APPLICATIONS MEMORANDA

1 December 1959

TECHNICAL APPLICATIONS MEMORANDA 1 December 1959

* NO.	TITLE	PURPOSE
1	Manual Methods for Converting G-15 Commands and Numbers in Decimal and Sexadecimal representation	To aid the operator in direct communication with the computer
2	Procedure to be followed when Issuing a Subroutine	To standardize the form of a subroutine and to ensure that complete information is made available to all concerned
3	Procedure to be Followed when issuing the description of a Project	To serve a recommended guide for writing descriptive information on programs
4	Notes on "Select Command Line and Return" - L_K N C 20 31	To clarify the use of L_K N C 20 31
14	Precautionary Stop DA-1 Command for all G-15D Loaders	To Assure the operator that the DA-1 Accessory is not operative during normal G-15 computation
17	Comparison of Programming - DA-1 and G-15	To illustrate different methods of programming the same problem using the DA-1 and the G-15
18	INTERCOM Timing (101)	To provide information on INTERCOM Timing
19	Cornu Spiral Plot	DA-1 - Plotter Demonstration
20	Condensed Format Tape Duplicator (Self-Loading)	To provide preliminary information on a service routine
21	DA-1 Timing and Control	Functional Explanation of DA-1
22	Flow Charting Routine	To provide a neat system of listing the flow of a routine

* NO.	TITLE	PURPOSE
23	Combined G-15D/DA-1 Computations Using DAPPER-1	To describe procedures for employing G-15D machine language program with DAPPER-1 for combined G-15D/DA-1 computations
24A	Intercom 1000 (24A)	Superseded by Intercom 1000, however, can be used for routines developed under this system.
24A Supplement 1	Intercom 1000	Listing of subroutines and additional information to 24A
25	Flow Diagramming for the G-15D Computer	To introduce the Bendix Template and to indicate some rules to follow when using the template
28	DAPPER 1-A	Description of revision to DAPPER-1
29	Programming and Operational Differences between the G-15A and G-15D	To summarize the differences between the two G-15 models for the benefit of the user of one machine who receives programs written for the other, or for the G-15A user who contemplates the acquisition of a G-15D
30	Changes in G-15 Logic - Shift Command	To provide for more flexibility in programming
32	Fixed Point, Double Precision Subroutines	To provide preliminary information on Double Precision, Fixed Point Subroutines
33	Programming Output to Punched Tape	To provide information concerning formats and timing on programs for punching leader and data on tape
39	Precession of Line 19	A method for programming a four-word precession in line 19
39 Supplement 1	"Block" Precession of Line 19	(1) To precess faster than normal precession speeds (2) To permit calculation during precession
40	The use of the Metalized Version of Mylar Tape	To prevent accidents

* NO.	TITLE	PURPOSE
41	Special Version of the "Select Command Line and Return to Mark Place" Command (L_K N C 20 31)	(1) To demonstrate the use of this command as an automatic loop counter (2) To demonstrate the use of this command as a many-way switch
41 Supplement 1	Additional Uses of the "Select Command Line and Return to Mark Place" Command (L_K N C 20 31)	(1) To demonstrate the use of this command as an automatic loop counter (2) To demonstrate the use of this command as a many-way switch
42	Correction of an Error in the Decimal to Binary Conversion in the Standard Program Preparation Routine	To avoid loss of information in line 03, u4 and u5 when using a "v" instruction
44	Miscellaneous Optional Modifications to the INTERCOM 1000 System	To provide advance preliminary information on optional changes that can be made to the INTERCOM 1000 system
45	PA-2 Plotter Calibration Routine	To provide an automatic means of checking the plotter calibration
46	Two-Word Registers	To specify conditions and describe methods by which a correct use of two-word registers may be made
47	Machine Language Subroutine and Regional Programming with POGO	To provide additional information on the preparation of machine language subroutines and to extend the scope of programming with POGO by the use of regional programming
48	Program DA-1 Halt when A Variable becomes zero	To present a method of programming to halt a DA-1 either for typeout for termination of a solution when any of specified variables becomes zero
49	Explanation due to Modification Memo #7	To provide information on changes in operating and programming techniques due to Modification Memo #7
50	Changes to PPR Lister	Required modification to PPR
51	Intercom Description and Addenda	To clarify the differences between existing Intercom 1000 D tapes
52	Changes to Intercom S. P. Appendix I Loader	To improve the performance of the automatic rewind feature of the Appendix I package

* Technical Applications Memorandums not listed are obsolete.

TECHNICAL APPLICATIONS MEMORANDUM NO. 1

9 April 1957

Title: Manual Methods for Converting G-15 Commands and Numbers in Decimal and Sexadecimal representation.

Purpose: To aid the operator in direct communication with the Computer.

Equipment affected: G-15 A and D

Effective Date: 9 April 1957

Instruction:

In order to properly understand the command structure of the G-15, it is advantageous to understand thoroughly the structure of commands and numbers within the computer.

All commands or numbers within the computer are expressed in binary form. However, all communication with the computer is expressed in sexadecimal numbers.

Since sexadecimal numbers are somewhat cumbersome to handle, the Program Preparation Routine has been created to provide a more rapid and convenient means of communicating with the computer. However, this routine does not completely relieve the programmer or computer operator of the responsibility of learning the details of the sexadecimal number system. Many situations will arise in which the programmer or operator may want to communicate directly with the computer. Without the use of conversion routines, input and output will be in the sexadecimal system. Therefore, it is necessary that the programmer understand and be the master of the sexadecimal number system.

Table VII of the Coding Manual denotes the relation of the decimal, sexadecimal, and binary number systems. This table will be beneficial in order to completely understand the conversion procedures explained below.

1) To convert a decimal command (T N C S D) to a sexadecimal number ($\pm\, d_1\, d_2 \ldots d_7$), proceed as follows:

Let Q = quotient and R = remainder

Then $\frac{D}{16} = Q_1 + \frac{R_1}{16}$ $\qquad d_7 = R_1$

$\frac{S}{8} = Q_2 + \frac{R_2}{8}$ $\qquad d_6 = 2R_2 + Q_1$

if $c < 4$ $\qquad d_5 = 4C + Q_2$

if $c > 4$ $\qquad d_5 = 4(C-4)+Q_2$ and make result negative [D]

$\frac{N}{16} = Q_3 + \frac{R_3}{16}$ $\qquad d_4 = R_3$

if no BP $\qquad d_3 = Q_3$

if BP $\qquad d_3 = Q_3 + 8$ [A]

$\frac{T}{16} = Q_4 + \frac{R_4}{16}$ $\qquad d_2 = R_4$ *

if Prefix = u, or no prefix and D = 31 $\qquad d_1 = Q_4$

if Prefix = w, or no prefix and D $\neq$ 31 $\qquad d_1 = Q_4 + 8$ [B]

* Exception:

When T = L + 1 (one larger than its location), D $\neq$ 31 and Prefix $\neq$ u:

if $c < 4$, add one to d_2 [C]

if $c \geqslant 4$, and T is even, add 2 to d_2 [D]

if $c \geqslant 4$, and T is odd, add one to d_2

Example:

Decimal Command u $\overset{T}{01}$ $\overset{N}{02}$ $\overset{C}{0}$ $\overset{S}{11}$ $\overset{D}{12}$ is to be located in 00.00

$$\frac{12}{16} = 0 + \frac{12}{16} \qquad d_7 = 12 = w$$

$$\frac{11}{8} = 1 + \frac{3}{8} \qquad d_6 = 2\,(3) + 0 = 6$$

$$d_5 = 4\,(0) + 1 = 1$$

$$\frac{02}{16} = 0 + \frac{2}{16} \qquad d_4 = 2$$

$$\frac{01}{16} = 0 + \frac{1}{16} \qquad d_2 = 1$$

$$d_1 = 0$$

Converted command = 010216w

Additional Examples:

	L	P	T	N	C	S	D	BP	Converted Command
A	00	u	01	02	0	11	12	-	018216w
B	00		05	06	0	11	12		850616w
C	00		01	02	0	11	12		020216w
D	01		02	04	4	11	12		-040416w

2) To Convert a sexadecimal number ($d_1 d_2 \ldots d_7$) to a decimal command, proceed as follows:

$$\frac{d_6}{2} = Q_6 + \frac{R_6}{2} \qquad D = 16\,R_6 + d_7$$

$$\frac{d_5}{4} = Q_5 + \frac{R_5}{4} \qquad S = 8\,R_5 + Q_6$$

if $(d_1\, d_2 \ldots d_7) > 0$ $\qquad C = Q_5$

if $(d_1\, d_2 \ldots d_7) < 0$ $\qquad C = Q_5 + 4$

[C]

if $d_3 \leq 6$ N = 16 d_3 + d_4 and there is no BP

if $d_3 > 6$ N = $16(d_3-8)+d_4$ and command contains BP [A]

if $d_1 \leq 7$ T = 16 d_1+d_2 and command is immediate

if D = 31, no Prefix
if D ≠ 31, Prefix = u *

if $d_1 > 7$ T = $16(d_1-8)+d_2$ and command is deferred

if D ≠ 31, no Prefix
if D = 31, Prefix = w [B]

* Exception:

When D ≠ 31 and command is immediate

if c < 4 and T = L + 2, subtract one from T
if c ≥ 4, L is odd and T=L+3, subtract 2 from T [C]
if c ≥ 4, L is even and T=L+2, subtract one from T

Example:

Location 00 contains 0202169

$\frac{6}{2} = 3 + \frac{0}{2}$ D = 16·0 + 9 = 09

$\frac{1}{4} = 0 + \frac{1}{4}$ S = 8·1 + 3 = 11

C = 0
N = 16·0 + 2 = 02
T = 16·0 + 2 = 02

Since T = L + 2, D ≠ 31 and C < 4,
T - 1 = 02 - 1 = 01

Decimal Command = 01 02 0 11 09

Additional Examples:

	L	Sexadecimal Number	P	T	N	C	S	D	BP
A	00	018216w	u	01	02	0	11	12	-
B	00	850616w		05	06	0	11	12	
C	01	-040416w		02	04	4	11	12	

3) To convert a decimal number $(D_1\ D_2\ \ldots\ D_k)$ to a sexadecimal number $(d_1\ d_2\ \ldots\ d_k)$, proceed as follows:

a) Decimal Integer $(D_1\ D_2\ \ldots\ D_k) = N$

$$\frac{N}{16} = Q_1 + \frac{R_1}{16} \qquad d_k = R_1$$

$$\frac{Q_1}{16} = Q_2 + \frac{R_2}{16} \qquad d_k = R_2$$

$$\vdots \qquad \vdots \qquad \vdots$$

Where $Q_i < 16$ $\quad \frac{Q_i}{16} = 0 + \frac{R_i}{16} \qquad d_i = R_i$

Example: $32580 = N$

$$\frac{N}{16} = 2036 + \frac{4}{16} \qquad d_k = 4$$

$$\frac{2036}{16} = 127 + \frac{4}{16} \qquad d_{k-1} = 4$$

$$\frac{127}{16} = 7 + \frac{15}{16} \qquad d_{k-2} = z$$

$$\frac{7}{16} = 0 + \frac{7}{16} \qquad d_1 = 7$$

Sexadecimal integer = 7z44

b) Decimal Fraction $(D_1\ D_2\ \ldots\ D_k)$

Let I = integral part of product
f = fractional part of product

$$(D_1\ D_2\ \ldots\ D_k)\cdot 16 = I_1 + f_1 \qquad d_1 = I_1$$

$$f_1 \cdot 16 = I_2 + f_2 \qquad d_2 = I_2$$

$$\vdots \qquad \vdots \qquad \vdots \qquad\qquad \vdots \quad \vdots$$

$$f_6 \cdot 16 = I_7 + f_7 \qquad d_7 = I_7$$

Summarizing this procedure, the fraction is multiplied by sixteen (16). Should the product of this multiplication be greater than one, $d_1 = I_1$. If the product is less than one, $d_1 = 0$. The fractional portion of the product is multiplied by sixteen (16) and the product is used to arrive at d_2 in the same way as d_1 is found above. This procedure can be repeated to compute the desired number of digits.

Example: Fraction = .4235

.4235 x 16 = 6 + .7760	d_1 = 6
.776 x 16 = 12 + .416	d_2 = w
.416 x 16 = 6 + .656	d_3 = 6
.656 x 16 = 10 + .496	d_4 = u
.496 x 16 = 7 + .936	d_5 = 7
.936 x 16 = 14 + .976	d_6 = y
.976 x 16 = 15 + .616	d_7 = z

Sexadecimal fraction = .6w6u7yz

4) To convert a sexadecimal number ($d_1\ d_2 \ldots d_k$) to a decimal number ($D_1\ D_2 \ldots D_k$), proceed as follows:

a) Sexadecimal integer ($d_1\ d_2 \ldots d_k$)

$d_1 \cdot 16 = P_1$

$(P_1 + d_2) \cdot 16 = P_2$

$(P_2 + d_3) \cdot 16 = P_3$

$(P_{k-2} + d_{k-1}) \cdot 16 = P_{k-1}$

$P_{k-1} + d_k = P_k$ = decimal integer

Example:

Convert 1z5y9 to decimal

1 • 16 = 16
(16 + 15) • 16 = 496
(496 + 5) • 16 = 8016
(8016 + 14) • 16 = 128480
128480 + 9 = 128489 = decimal integer

b) Sexadecimal fraction ($d_1\ d_2\ .\ .\ .\ .\ d_k$)

$$\frac{d_k}{16} = Q_1$$

$$\frac{d_{k-1} + Q_1}{16} = Q_2$$

$$\frac{d_2 + Q_{k-2}}{16} = Q_{k-1}$$

$$\frac{d_1 + Q_{k-1}}{16} = Q_k = \text{Decimal fraction}$$

Example:

Convert .9x7u to decimal (7 places)

$$\frac{10}{16} = .625$$

$$\frac{7.625}{16} = .4765625$$

$$\frac{12.4765625}{16} = .84228516 \; *$$

$$\frac{9.84228516}{16} = .61514282$$

Decimal Fraction = .6151428

* Round to one more place than desired in converted number.

Note that multiplying by .0625 is equivalent to dividing by 16.

Action By: All personnel concerned

References: G-15 A and D Coding Manuals

Remarks: None

Prepared by: D. E. Hassell *[signature]*

Approved By: T. Yamashita *[signature]*

TECHNICAL APPLICATIONS MEMORANDUM NO. 2

1 December 1959

TITLE: Procedure to be followed when Issuing a Subroutine

PURPOSE: To standardize the form of a subroutine and to ensure that complete information is made available to all concerned

EQUIPMENT AFFECTED: All

EFFECTIVE DATE: 1 December 1959

INSTRUCTION:

A subroutine shall consist of the following parts - each part being presented on a separate sheet:

1. Detailed description
2. Specification sheet
3. Check sums
4. Method sheet (if applicable)
5. Flow Chart
6. Memory Allocation Sheet
7. Coding sheets

The method of preparation of each of the above items shall be as follows:

1. Detailed Description: Each description shall begin with the standard heading, which will include the title of a subroutine and the number. (The number will be allocated by the Applications Section.) It will also include the page number, the date, the line number, and by whom it was prepared and approved.

   It should then be followed by stating the nature of the subroutine, its purpose, its limitations, etc. Should there be any doubt in its execution, a simple example of its particular application should be given.

   It may happen that the subroutine must operate in conjunction with some other subroutines. This must then be specifically stated and the storage locations of supplementary subroutines be given.

If the subroutine is based on a mathematical formula or method, reference to the method sheet should be made.

Service Routines should also include information on the input format, output format, punched tape input, punched tape output, magnetic tape (if applicable), etc.

2. Specification Sheet: The specification sheet shall be of a standard form as shown in Appendix "A" of this memorandum.

3. Check Sums: A set of hexadecimal check sums should be listed with proper notation of each, i.e., loader, line 1, line 4, etc.

4. Method Sheet: Whenever applicable, the method sheet should be attached to the subroutine. Inasmuch as it is difficult to standardize its form, the minimum information should include:

   a. Statement of mathematical formulae
   b. Approximating polynomial and the number of terms
   c. List of coefficients in decimal and sexadecimal notation
   d. Error Term (and therefore the accuracy expected)
   e. Computational method and rearrangement of terms (if any)
   f. References
   g. Remarks

5. Flow Chart: Pending the issue of the Bendix Computer Template, the symbols recommended by the Users' Conference should be followed. The complete method of flow charting will be outlined in the forthcoming Technical Applications Memorandum on the subject.

6. Memory Allocation Work Sheet: The Memory Allocation Work Sheet was recently introduced as a supplement to coding sheets. In the light of recent recommendations at the Users' Conference, the general form of the work sheet will be somewhat modified and the complete description issued in the forthcoming Technical Applications Memorandum on the subject.

7. Coding Sheets: Coding sheets should be completed in the usual manner, but the following conventions were found to be helpful:

   a. Constants

   Memory location only should be entered in the space provided for commands. The space itself should be enclosed by two vertical lines followed by a sexadecimal type-out of a constant in the "Notes" column.

b. <u>Command Constants</u>

Command constants should be entered in the space provided. The space itself should be enclosed by two vertical lines.

c. <u>Variable Commands</u>

Variable commands should be entered in the usual manner and should be enclosed by symbolic representation of two acute angles.

For the illustration of the above conventions see Appendix "B".

ACTION BY: All personnel concerned

REFERENCES: Users' Conference Proceedings

REMARKS: None

PREPARED BY: Z. Jelinski

CHECKED BY:

APPROVED BY: T. Yamashita

ZJ:TY:gm

SPECIFICATIONS

	Explanation of Terms
TYPE:	State whether subroutine, service routine, sorting routine, etc.
EQUIPMENT AFFECTED:	State whether G-15A, G-15D, DA-1, etc.
MODE OF OPERATION:	State whether { Fixed point - single precision, double precision; Floating point - single precision, double precision }
EXECUTION:	State command line from which it is to be executed
ENTRY:	State entry conditions, i.e., memory location and return command location
SCALING:	State binary scaling complete
DATA INPUT:	State relevant memory locations
DATA OUTPUT:	State relevant memory locations
EXECUTION TIME:	State time for one iteration
ERROR STOPS:	
ALL STORAGE LINES USED:	State long and short lines used for temporary storage
	State short lines available for use
SUMMATION CONSTANT:	State memory location and the summation constant

REMARKS:

- SAMPLE -

MEMORANDUM NO. 2

APPENDIX "B"

Bendix Computer

Los Angeles 45, California

Page ___ of ___

Date: ___________

Prepared by ____________________

Line ___________

G-15D
PROGRAM PROBLEM: ______________________________

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35
36	37	38	39
40	41	42	43
44	45	46	47
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63
64	65	66	67
68	69	70	71
72	73	74	75
76	77	78	79
80	81	82	83
84	85	86	87
88	89	90	91
92	93	94	95
96	97	98	99
u0	u1	u2	u3
u4	u5	u6	

L	P	T or L_k	N	C	S	D	BP	NOTES
57								c_1 = -zzzzzzz Extractor
76		77	86	1	21	31		Command Constant
10	<	11	35	1	02	28	>	Variable Command

FORM BCD 108-5.0

TECHNICAL APPLICATIONS MEMORANDUM NO. 3

1 December 1959

TITLE: Procedure to be Followed When Issuing the Description of a Project

PURPOSE: To serve as a recommended guide for writing descriptive information on programs

EQUIPMENT AFFECTED: All

EFFECTIVE DATE: 1 December 1959

INSTRUCTION:

I. All completed programs shall be issued under a project number assigned by the Applications Section. In general the following categories of projects shall be used:

1. Demonstration: These are projects specifically designed to demonstrate the computer and shall be absolutely contained in themselves, namely input data specifically "tailored" to suit the project will be treated as the integral part of the project.

2. General: This group of projects will encompass all problems designed for a specialized field which would not be adaptable to any other field. If mathematical representation is used, this would usually consist of one or more specific equations, but their solution would not necessarily lend itself to be generalized.

II. Bendix Computer Users' Exchange Organization

This memorandum can be used by the users as a guide when contributing a program to the Exchange Library. For the category assignment, refer to the Administrative Codes and the By-Laws of the G-15 Users' Exchange Organization.

III. Preparation of the Project

A complete project will consist of the following parts:

1. Front page
2. Table of Contents
3. Check sums
4. Part I, Introduction
5. Part II, Problem Statement
6. Part III, Computational Approach
7. Part IV, Operating Instructions
8. Part V, Flow Charts and Coding Sheets
9. Part VI, References
10. Part VII, Sample Numerical Input or Output Data
11. One page abstract of the program

The front page shall consist of:

1. Heading
2. Title - a complete title of the project should be given
3. Type - type of project should be given, e. g., General
4. Category
5. Equipment Affected, e. g., G-15D, CA-2
6. Subroutines Used
7. Prepared by
8. Checked by
9. Approved by
10. Date
11. Revision (if any)

For the layout of this page, see Appendix "A".

For the Table of Contents, see Appendix "B".

Check Sums: A set of hexadecimal check sums should be listed with proper notation of each, i.e., loader, line 1, line 4, etc.

Introduction: A short outline of the problem, including the purpose, acknowledgments, circumstances leading to its inception, etc., should be here stated.

A detailed Problem Statement should be outlined. This should follow by mathematical formulae (if any), equations to be solved, parameters used and their limitations.

Next, the approximation polynomial should be stated, including the list of coefficients, the number of terms used, the error term and therefore, the accuracy expected within specified significant digits. Reference should be made to sources of obtaining the approximating polynomial.

Computational Approach:

Rearrangement of terms of the polynomial (if any) for computational ease should be stated and this should be followed by a step-by-step description.

At this stage, if practicable, a sample solution should be given. Scaling or information on Floating Point, Single or Double Precision, Memory Utilization, i.e., the total number of lines used for program and constants should be contained in this section.

Operating Instructions:

This section should contain:

1. Starting the Problem - instructions regarding punched tape magazines, typewriter operation, magnetic tape operating (if applicable), DA-1, AN-1, CA-1, etc.

2. Input Instructions - type-in instructions, including format, number of digits, etc.

3. Output Instructions - including format with headings, subheadings, abbreviations, etc.

4. Execution Time - for one iteration, for the complete problem separating input and output operating from the actual time of computation.

5. Error Stops - including procedure to continue the problem after an error stop.

Flow Charts and Coding Sheets (self explanatory)

References: All references made should be listed.

Numerical Data: All projects should list:

1. Sample Input Data
2. Sample Output Data

Abstract of a Project - should contain enough information to determine the main problem solved and the program limitations.

ACTION BY: All personnel concerned

REFERENCES: Administrative Code of the G-15 Users' Exchange Organization
By-Laws of the G-15 Users' Exchange Organization

REMARKS: This memorandum supersedes Technical Applications Memorandum No. 3 dated 11 April 1957 and No. 3A dated 9 May 1958.

PREPARED BY: T. A. Blanke

CHECKED BY:

APPROVED BY: T. Yamashita

ZJ:TY:gm

MEMORANDUM NO. 3 - SAMPLE - APPENDIX "A"

BENDIX COMPUTER DIVISION OF BENDIX AVIATION CORPORATION
5630 Arbor Vitae Street, Los Angeles 45, California

APPLICATIONS SECTION

Project No.

TITLE:

TYPE:

CATEGORY:

EQUIPMENT AFFECTED:

SUBROUTINES USED:

PREPARED BY:

CHECKED BY:

APPROVED BY:

DATE: Revision:

Project No.

TABLE OF CONTENTS

Check Sums

.0000000	Loader
-.zyx0yu6	Line 15
-.xu0uv60	Line 13
-.y9055x3	Line 12
-.774u4v8	Line 11
-.vxz6857	Line 10
.vxz516x	Line 09
.4uw3zz7	Line 08
-.7658z0u	Line 07
-.w2v6x3u	Line 06
-.4yu2y3v	Line 04
-.y17y2w9	Line 03
.3zy7790	Line 02
.10w0985	Line 01
-.0x74923	Line 00

TECHNICAL APPLICATIONS MEMORANDUM NO. 4

11 April 1957

TITLE: Notes on "Select Command Line and Return" - L_k N C 20 31

PURPOSE: To clarify the use of L_k N C 20 31 command

EQUIPMENT AFFECTED: G-15D

EFFECTIVE DATE: 11 April 1957

INSTRUCTION:

Generally, this command is used as a subroutine exit command, the next command being taken from word T (T having been specified by a previous "Mark" command) of Line C. When so used, the command is written:

$$L_2 \; L_1 \; C \; 20 \; 31$$

In special applications it may be desired to take the next command from word N of Line C, rather than the "Marked" T. In this case the command would be

$$L_k \; N \; C \; 20 \; 31 \qquad \text{where } L_k \leq N \leq T$$

It should be noted that this command can not be used as a stopping place in break point operation. Also, in single-cycle operation, this command and the one following are executed. Further, in single-cycle operation, the special application noted above fails, with the next command taken from the "Marked" T.

ACTION BY: All personnel concerned

REFERENCES: None

REMARKS: None

PREPARED BY: R. J. Margolin
APPROVED BY: T. Yamashita

RJM/TY/bg

TECHNICAL APPLICATIONS MEMORANDUM NO. 14

23 April 1957

TITLE: Precautionary Stop DA-1 Command for all G-15D Loaders

PURPOSE: To assure the operator that the DA-1 accessory is not operative during normal G-15D computation

EQUIPMENT AFFECTED: All programs for G-15D equipment

EFFECTIVE DATE: 23 April 1957

INSTRUCTION: The stop DA-1 command is of the form

	00	N	1	19	31 or
w	u7	N	1	19	31.

When the turn on cycle has been completed, there is a random possibility that the DA-1 is in the ON condition. This prevents a portion of the G-15D memory from operating for G-15D computing. Insertion of either of the above commands assures that the DA-1 is OFF and that normal computation may proceed. The stop DA-1 command must be active during word u7. It is imperative that this command be included in <u>all</u> loaders whether DA-1 is part of an installation or not.

ACTION BY: All personnel concerned

REFERENCES: G-15D Coding Manual
DA-1 Operating Instructions

REMARKS: None

PREPARED BY: C. W. Armstrong

APPROVED BY: T. Yamashita

TECHNICAL APPLICATIONS MEMORANDUM NO. 17

24 May 1957

TITLE: Comparison of Programming - DA-1 and G-15

PURPOSE: To illustrate different methods of programming the same problem using DA-1 and G-15

EQUIPMENT AFFECTED: DA-1 and G-15

EFFECTIVE DATE: 24 May 1957

INSTRUCTION:

Probably the greatest advantage of the DA-1 in this example is the comparative ease of programming. This advantage carries over in most cases. The time necessary for solution on a general purpose computer depends not only on the precision required, but also on the method employed and the complexity of the problem. On the other hand, all 108 integraters are iterated on each drum revolution in the DA-1 so that an increase in complexity of the problem does not necessarily mean more time is required for solution. The interval length (and hence the number of steps) is, on both types of machines, a function of the precision requirements. While a sophisticated integration technique on a general precision computer will permit the use of fewer steps, each step will require more computation time (how much more depends on the problem) than a step on a DA-1.

The combination of a general purpose computer and a DA-1 should prove to be very powerful. For example, programmed modification of boundary values is possible, also, the full input-output facilities of the general purpose machine are available to the DA-1.

For illustration see Apendices "A" and "B".

ACTION BY: All personnel concerned

REFERENCES: None

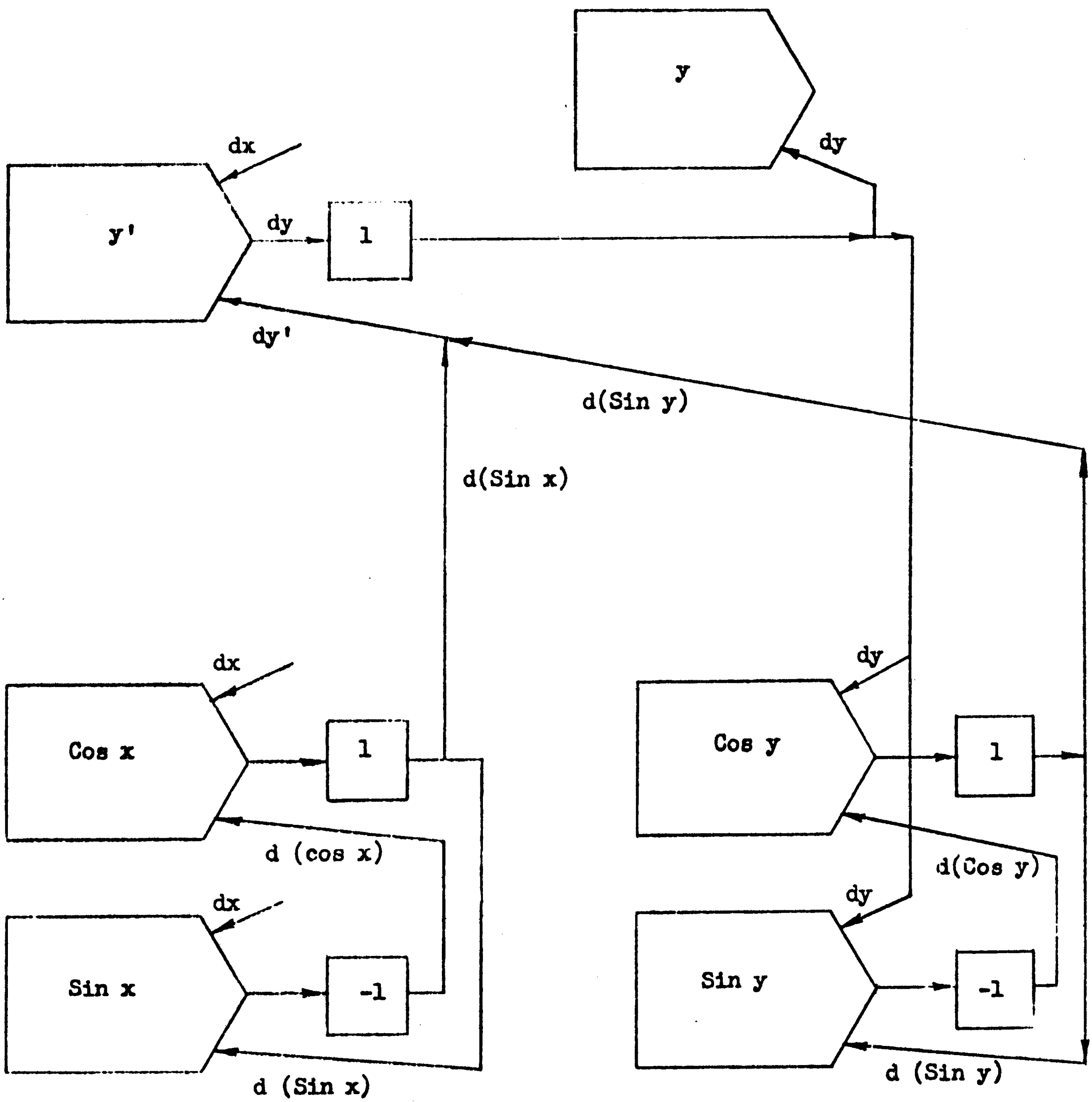
REMARKS: None

PREPARED BY: D. Stein Dale W Stein

APPROVED BY: T. Yamashita T. Yamashita

Consider the equation y' = Sin x + Sin y.

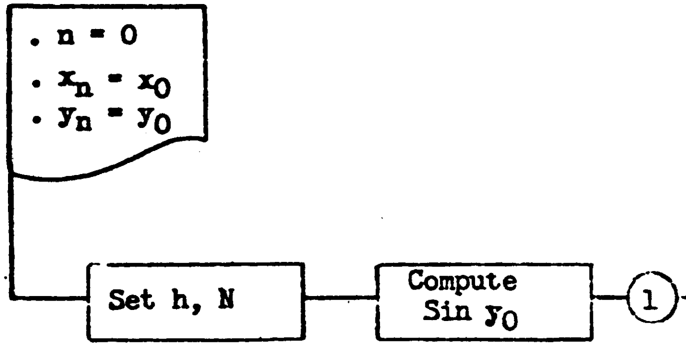
Solution of this equation on the G-15D - DA-1 combination involves the use of 6 integrators connected as below:

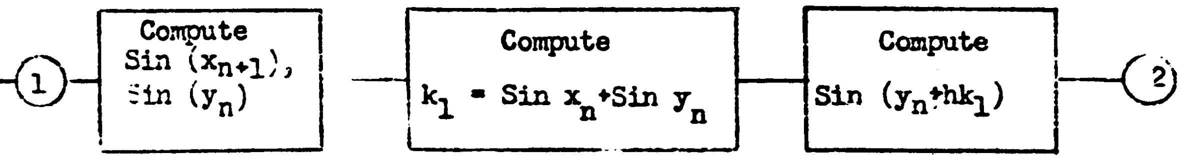


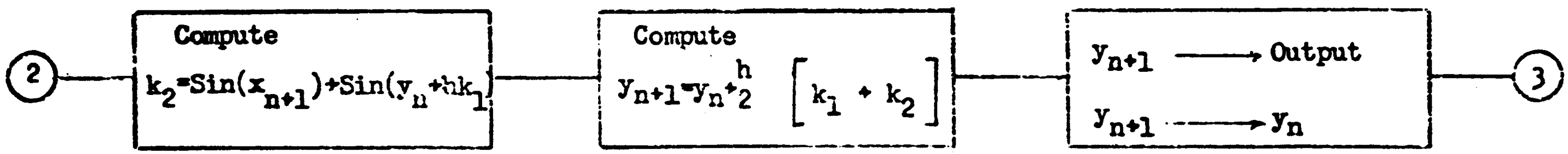
y' = Sin x + Sin y

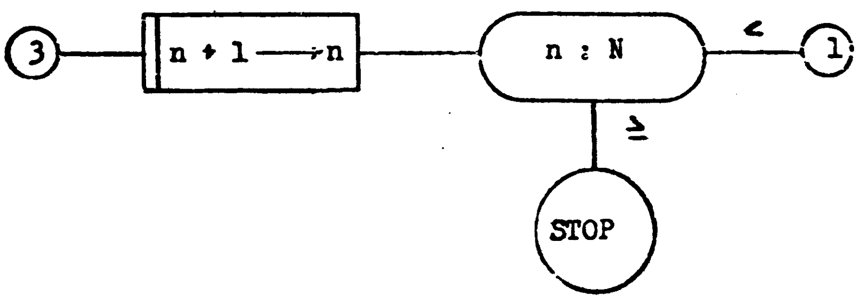
d(y') = d (Sin x) + d (Sin y)

The same problem could be solved on a General Purpose Computer in any one of a number of ways. The following is perhaps representative:









y' = Sin x + Sin y

Use Simplified Runge-Kutta Method:

$$y_{n+1} = y_n + \frac{h}{2} \left[k_1 + k_2 \right] + 0(h^3)$$

Where: k_1 = Sin x_n + Sin y_n; k_2 = Sin x_{n+1} + Sin$(y_n + hk_1)$

TECHNICAL APPLICATIONS MEMORANDUM NO. 18

6 June 1957

TITLE: INTERCOM Timing (101)

PURPOSE: To provide information on INTERCOM Timing

EQUIPMENT AFFECTED: Not Applicable

EFFECTIVE DATE: 6 June 1957

INSTRUCTION:

The following timing consideration when using INTERCOM was submitted by Dr. D. Peaceman of Humble Oil Company. It is suggested that all G-15 users study these considerations to minimize computing time when coding in INTERCOM.

Computing time is minimized if orders are stored in locations 000 to 247 (inclusive), and if numbers are stored in the locations given in the following table (note that the optimum addresses vary somewhat for the different orders):

ORDER		ADDRESS
4v	(Clear and Add)	408 - 599
49	(Clear and Subtract)	408 - 583
5v	(Add)	408 - 727
59	(Subtract)	408 - 711
67	(Multiply)	408 - 823
4z	(Divide)	408 - 631
5x	(Store)	472 - 695

Whenever inconvenient, these restrictions may be violated without causing errors in computations.

The time of execution of the various commands is always an integral number of machine cycles, where each machine cycle is approximately 29 milliseconds. When the above restrictions are obeyed, the STORE (5x) command takes one cycle; all the others listed above take two cycles. Violation of the restrictions given above each adds one cycle to all of these commands; the use of address modification by an index register adds an additional cycle.

Example: CA, 4v5230 takes two cycles,
4v5002 takes three cycles,
4v7003 takes four cycles.

The time of execution of other orders, if they are located within 000 to 247 (inclusive), are as follows:

ORDER		NO. OF CYCLES
13 (Absolute Value)		2
29 (Negate)		2
07 (Square Root)		6
11 (Ln)		11
16 (e^x)		12
44 (Return to mark place) with abc=500		1
3x (Set B Register)		3
40 (Set D Register)		3
41 (Set L Register)		3
1y (Replace Index Register)		6
22 (Type the Stack)	If not Output limited	10 - 11
1z (Type and Tab)	If not Output limited	10 - 11
20 (Type and Carriage Return)	If not Output limited	10 - 11
33 (Ring Bell) with abc=000		3

The time of execution of transfer and conditional transfer orders depends on the effective address, abc + B(k), of the order, whether address modification is used, and whether the condition for transfer is met. The table below indicates the optimum effective addresses and the minimum execution time of the various transfer orders:

ORDER	OPTIMUM ADDRESS	MIN. NO. OF CYCLES
19 (Unconditional Transfer)	000 - 199	1
10 (Transfer if A Register ≥ 0)	000 - 127	1
0v (Transfer if A Register < 0)	000 - 087	1
43 (Mark place and transfer)	000 - 864	2
65 (Increment B Register)	000 - 864	4
57 (Decrement B Register)	000 - 864	4

Failure to use an optimum address, use of address modification, and failure of the condition for transfer to be met, each add an additional cycle.

ACTION BY: All personnel concerned

REFERENCES: None

REMARKS: The above information has been sent to all G-15 users.

PREPARED BY: T. Yamashita
APPROVED BY: T. Yamashita T. Yamashita

21 June 1957

TECHNICAL APPLICATIONS MEMORANDUM NO. 19

TITLE: Cornu Spiral Plot

PURPOSE: DA-1 - Plotter Demonstration

EQUIPMENT AFFECTED: G-15D, DA-1, and PA-1

EFFECTIVE DATE: 21 June 1957

INSTRUCTION:

This problem is programmed in conjunction with Dapper-1.* The program as listed on the accompanying sheet generates a spiral in the first quadrant. To generate a spiral in the third quadrant, change the sign of the y_0 value of integrator 00, i.e., type (using Dapper-1) -50.9999999 Tab 50 24 00 1 Tab s and proceed as before. The paper on the plotter should be at least 11 1/2" in x direction and 10 1/2" in y direction if both branches are to be plotted. Plotting time is approximately 10 minutes per branch with manual stop when the spiral begins to converge. Integrator 04 will overflow after a somewhat longer time thereby stopping the plotter.

To operate follow instructions in Table IV, page 71 of the DA-1 Manual with the following addenda:

a. In Step 4, set switches for 108 integrator operation.

b. In Step 8, type 4 Tab s (for 108 integrator operation).

c. In Step 9, type program from accompanying program sheet.

After Step 10, set up plotter for plotting spiral, proceed.

* DA-1 Program Preparation Routine

CORNU SPIRAL - FAST PLOT PROGRAM USING DAPPER-1

```
00  04  3  Tab  s
04  08  3  Tab  s
04  12  3  Tab  s
00  13  3  Tab  s
00  17  3  Tab  s
13  03  3  Tab  s
17  02  3  Tab  s

00  04  4  Tab  s
08  12  4  Tab  s
08  13  4  Tab  s
12  08  4  Tab  s
12  17  4  Tab  s

 51.3141593  Tab  52  08  2  Tab  s
-51.3141593  Tab  52  12  2  Tab  s
 50.3333333  Tab  50  13  2  Tab  s
 50.3333333  Tab  50  17  2  Tab  s

50.9999999  Tab  50  24  00  1  Tab  s
46.0000000  Tab  53  38  04  1  Tab  s
51.1000000  Tab  51  44  08  1  Tab  s
46.0000000  Tab  51  44  12  1  Tab  s
46.0000000  Tab  51  44  13  1  Tab  s
51.1000000  Tab  51  44  17  1  Tab  s
```

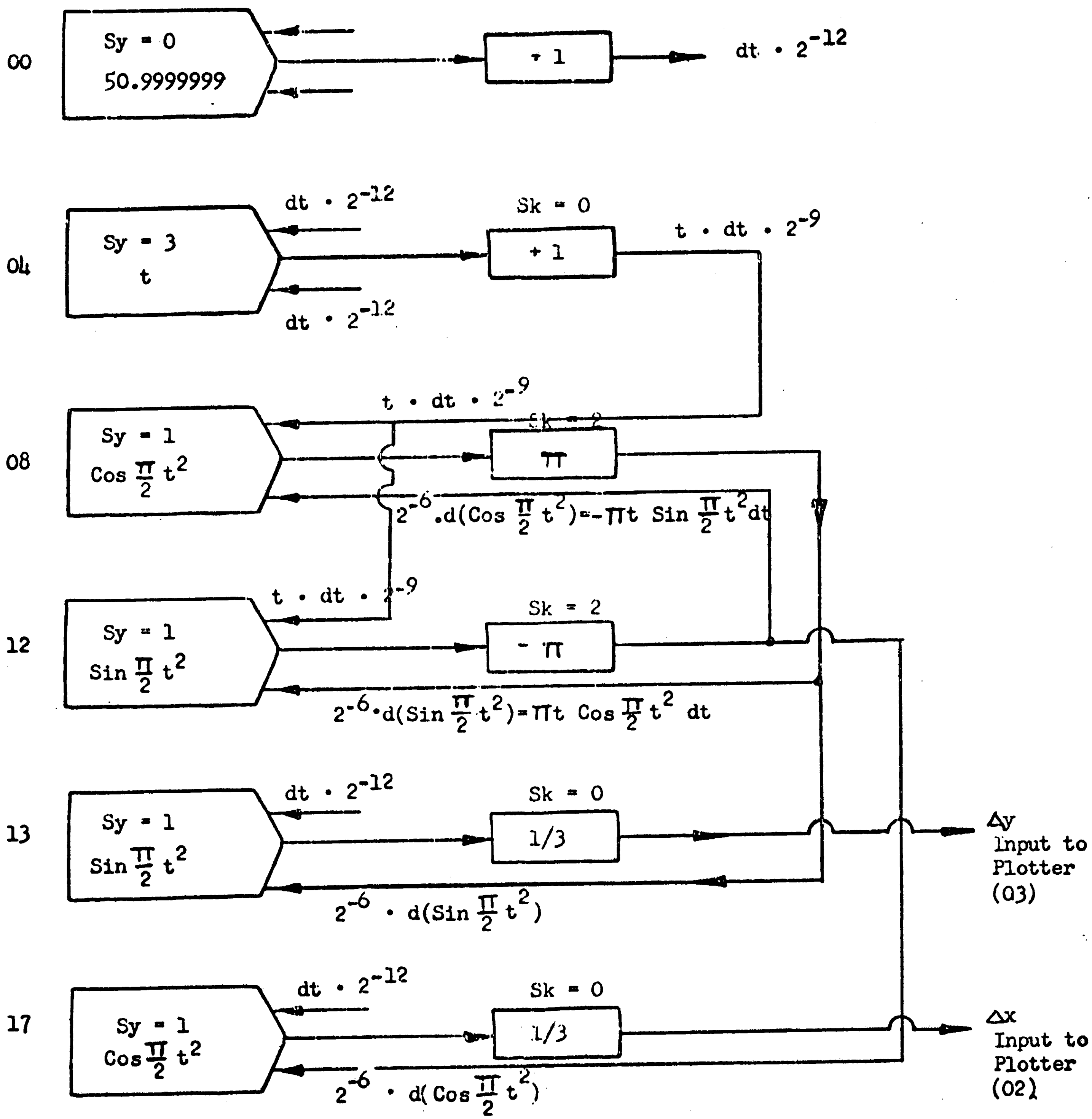
ACTION BY: All Personnel Concerned

REFERENCES: DA-1 Manual

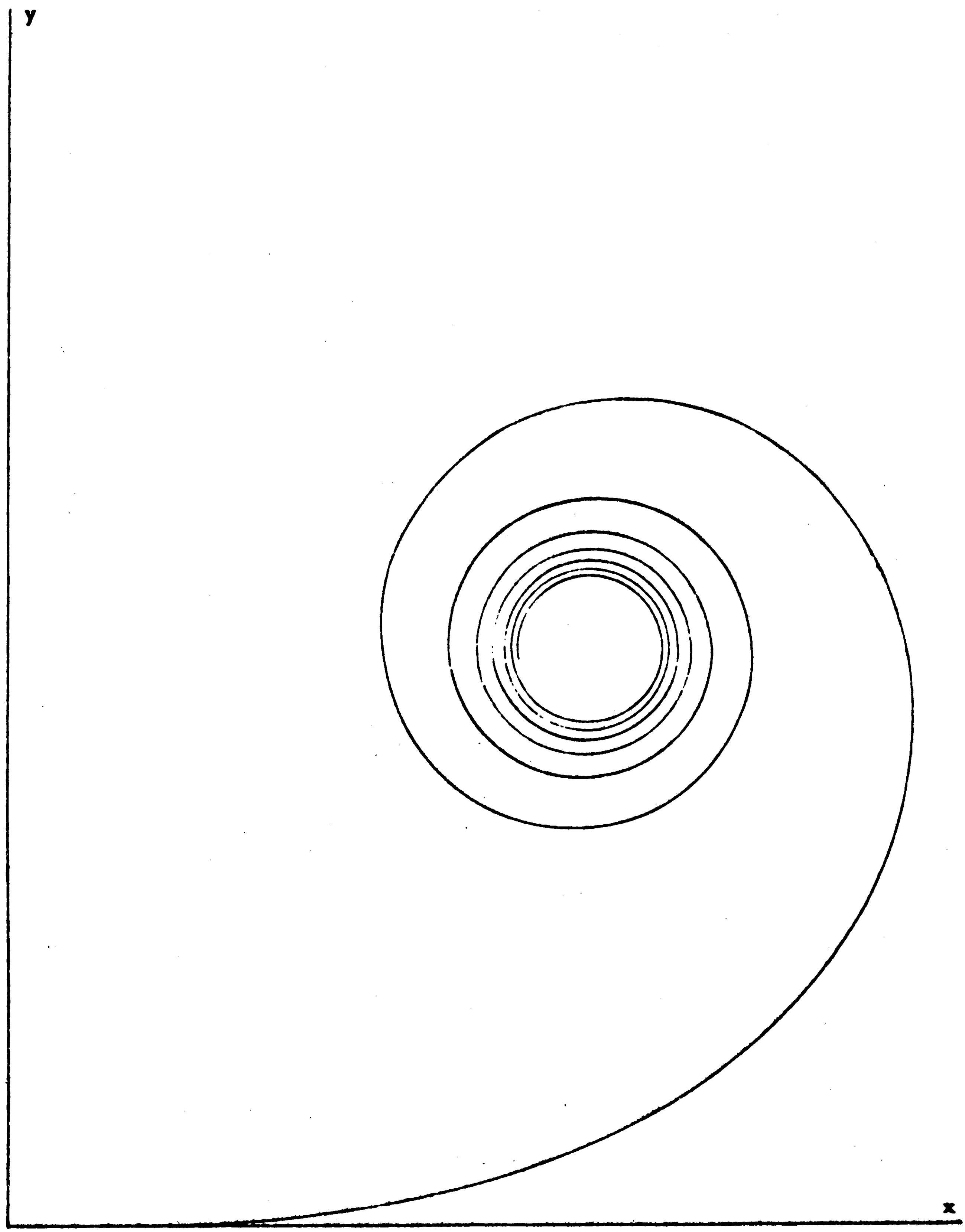
REMARKS: None

PREPARED BY: D. Stein
APPROVED BY: T. Yamashita

CORNU SPIRAL - FAST PLOT FLOW CHART



THE CORNU SPIRAL


y
x

TECHNICAL APPLICATIONS MEMORANDUM NO. 20

18 July 1957

TITLE: Condensed Format Tape Duplicator (Self loading)

PURPOSE: To provide advance information on a service routine

EQUIPMENT AFFECTED: G-15D

EFFECTIVE DATE: 18 July 1957

INSTRUCTION:

Description:

This is a service routine for G-15D designed to duplicate tapes, having the output tape punched with a format consisting of 29 digit codes and an end (reload) code. This represents a saving of 7 tape characters per four words, or 18.9% reduction in tape length between 7 1/2 inch leaders, compared with "standard" format. Not only is tape material conserved, but tapes punched in the condensed format can be read and reversed more rapidly (e.g. INTERCOM). For efficiency, short blocks are processed before punching, during typeout of the check sum.

The complete program, including its own loading routine, is stored in words 00 - 47 of line 01 for execution. The first loading command (word 00) is replaced after loading, so the "f" key may be used to reenter the program from the beginning.

Operation:

A. Loading:

"p" key to read the self loading program tape, then COMPUTE to either GO or BP. A halt occurs to permit placement of the tape to be duplicated.

If the program tape is loaded by PPR instruction w00 (tab) s, then type 1 x 03 (tab) s, followed by entry with 104 x 04 (tab) s.

B. Compute to GO:

After line 19 is cleared a block of tape is read, summed in AR, and copied into 18. Line 19 is cleared again, and while the check sum is being typed from 19.u7, line 18 is given four-word precessions until words u4 - u7 are not all clear. After precession and typeout, while line 19 is clear, a 7 1/2 inch leader is punched, followed by setting of "Ready". Then line 18 is copied into 19 and punched in the format described above. At the conclusion of punching, the next tape block is read and the process repeated.

C. Compute to BP:

The commands to type the check sum and to punch the record on tape both have break points. If an incorrect check sum is typed after reading a tape being duplicated, the tape may be reversed with the "b" key and reread after "f". The position of the COMPUTE SWITCH should not be changed during the punching of leader. If it is desired to stop after reproducing a particular block of tape (e.g., the last one), the COMPUTE SWITCH should either be set to BP during check sum typeout, or turned off during the punching of a record when the indicator lights show source 28, destination 31.

D. Modifying Tape:

If a tape is to be reproduced with modifications, this program may be used jointly with PPR. After the BP halt on check sum typeout, the contents of the last block read are in line 18. Assuming PPR is stored, type "5f" with ENABLE on, then COMPUTE to GO or BP, and use PPR to modify line 18. To transfer control from PPR to this program for typeout of the new check sum and condensed format punching before more tape is read, use PPR instructions 146 x 04 (tab) s.

Use of Storage:

Line 01, words 00 - 47 (entire line disturbed on loading); line 02, words 00 - 03 (format for line 19 output); all arithmetic registers; line 23 during tape reading.

ACTION BY: All Personnel Concerned

REFERENCES: None

REMARKS: Appendix A - Specification Sheet

Appendix B - Memory Print Out

Appendix C - Coding Sheets

PREPARED BY: S. H. Lewis S.H. Lewis

APPROVED BY: T. Yamashita T. Yamashita

SHL/TY/jr

SPECIFICATIONS

TYPE: Service Routine (Complete Program; Self-loading)

EQUIPMENT AFFECTED:	G-15D
MODE OF OPERATION:	Not Applicable
EXECUTION:	Line 01
ENTRY:	To read tape, sum and punch: Word 00 To sum and punch (line 18): Word 46
SCALING:	None
DATA INPUT:	Punched tape, or line 18
DATA OUTPUT:	Punched tape
EXECUTION TIME:	Variable. Sum and punch full block from line 18: 54 seconds.
ERROR STOPS:	None
ALL STORAGE LINES USED:	01, 18, 19, 23, AR, ID, MQ, PN
SUMMATION CONSTANT:	.uzu364z
REMARKS:	See description of use.

CONDENSED FORMAT TAPE DUPLICATOR (SELF-LOADING)

MEMORY PRINT OUT

.uzu364z

320u2zz	2z2z253	8002381	-8228322
2x8715z	2w2w01z	12515z	292v253
29292xz	100000	2727w3x	242439z
8309382	2000000	232303u	9249248
21673zz	1725v	1z1z03w	1x1xu52
1w1wu52	1v1vu52	1u1uu52	u02103w
191933w	yv15393	179x13z	1516333
1414272	131367x	111139z	-8211302
-8w00000	1000000	9000000	-8y10038
8x2403w	w0w33w	-800v342	u0u1zz
70739z	707333	3062zz	62x03w
467z	40021z	3036vz	101261

Bendix Computer

Los Angeles 45, California

Page 1 of 2

Date: 7/18/57

Line 01

G-15D

Prepared by S. H. Lewis

PROGRAM PROBLEM: CONDENSED FORMAT TAPE DUPLICATOR (SELF LOADING)

~~0~~	~~1~~	~~2~~	~~3~~
~~4~~	~~5~~	~~6~~	~~7~~
~~8~~	9	~~10~~	11
~~12~~	13	~~14~~	~~15~~
~~16~~	~~17~~	~~18~~	~~19~~
~~20~~	~~21~~	~~22~~	23
24	25	26	27
28	~~29~~	~~30~~	~~31~~
32	33	34	35
36	37	38	39
40	41	42	43
44	~~45~~	46	47
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63
64	65	66	67
68	69	70	71
72	73	74	75
76	77	78	79
80	81	82	83
84	85	86	87
88	89	90	91
92	93	94	95
96	97	98	99
u0	u1	u2	u3
u4	u5	u6	

L	P	T or L_k	N	C	S	D	BP	NOTES
<00	u	01	01	0	19	01	>	(19) → 01
01		03	03	1	21	31		N. C. From 01
03		00	04	1	19	31		DA-1 Off
04		05	45	0	01	28		Clear Command → AR
05		03	06	0	23	31		Clear
45		00	02	0	28	01		(AR) → 01.00
02		04	00	0	16	31		Halt
<00		03	06	0	23	31>		Clear
06	u	07	07	0	25	19		0 = (ID) → 19
(07		07	07	0	28	31		Ready ?
(08		10	10	0	15	31		Read Tape
10		11	12	0	25	28		0 = (ID) → AR
12		14	16	4	01	24		Format → MQ
14								1000000
15								-8w00000
16		02	17	4	24	02		(MQ) → 20.02,03
17		17	17	0	28	31		Ready ?
18	u	19	19	1	19	29		Σ(19) → AR+
19	u	20	20	0	19	18		(19) → 18
20	u	21	22	0	25	19		0 = (ID) → 19
22		u7	21	0	28	19		(AR) → 19.u7
21		23	29	0	09	31	-	Type (19)
29		30	31	0	01	28		Test Command → AR
30	u	00	23	0	18	27		Test Command
31		33	u3	0	31	31		N. C. From AR

RM CD -5.0

Bendix Computer

Los Angeles 45, California

Page 2 of 2

Prepared by S. H. Lewis

Date: 7/18/57

G-15D

PROGRAM PROBLEM: CONDENSED FORMAT TAPE DUPLICATOR (SELF LOADING)

Line 01

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35
36	37	38	39
40	41	42	43
44	45	46	47
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63
64	65	66	67
68	69	70	71
72	73	74	75
76	77	78	79
80	81	82	83
84	85	86	87
88	89	90	91
92	93	94	95
96	97	98	99
U0	U1	U2	U3
U4	U5	U6	

L	P	T or L_k	N	C	S	D	BP	NOTES
								Test = 0
23		24	25	0	25	28		0 = (ID) → AR
24		32	33	0	01	28		≠ 0 Leader Format → AR
25	u	26	26	2	18	18		(18) —tva→ 18
26	u	27	27	2	18	18		(18) —tva→ 18
27	u	28	28	2	18	18		(18) —tva→ 18
28	u	29	29	2	18	18		(18) —tva→ 18
32								Leader Format = 9249248
33		34	35	0	01	26		Punch Format → PN
34								Punch Format = 2000000
35		03	09	0	28	02		(AR) → 02.03
09		00	11	4	26	02		(PN) → 02.00,01
11		13	36	0	01	28		C_1 → AR
13								C_1 = 9000000
36		36	36	0	28	31		Ready ?
37		38	39	3	01	29		C_2 ⇒ AR +
38								C_2 = 0100000
39		41	41	0	22	31		(T_1.AR) → Test
41		01	37	0	10	31		Punch Leader
42		44	44	0	00	31		Set Ready
44		02	40	4	25	02		0 = (ID) → 02.02,03
40	u	41	43	0	18	19		(18) → 19
43		45	07	0	10	31	-	Punch Tape
								Reentry From PPR
46	u	47	47	0	18	19		(18) → 19
47		50	10	0	23	31		Clear

FORM BCD 108-5.0

TECHNICAL APPLICATIONS MEMORANDUM NO. 21

15 July 1957

TITLE: DA-1 Timing and Control

PURPOSE: Functional Explanation Of DA-1

EQUIPMENT AFFECTED: G-15D, DA-1

EFFECTIVE DATE: 15 July 1957

INSTRUCTION:

The information on DA-1 Timing and Control is grouped under the following headings:

I. Introduction to DA-1 Timing and Control

II. The DA-1

A. Timing

1. The Integrand and Scaling

2. The R register and the Constant K

3. The arithmetic of dZ and K dZ outputs

4. The Timing and Addressing information

B. Control Considerations

1. G-15D Commands

2. Overflows and halts

3. Plotters and Plotter/follower

C. Communication between the G-15D and DA-1 -- physical characteristics

I. Indroduction to DA-1 Timing and Control

The material to follow will aid the programmer who wishes to write G-15D programs for combined G-15D/DA-1 operations. The internal logic of the DA-1 is presented in the course taught by the BCD Training Group. Knowledge of both G-15D programming and the use of DAPPER-1 are assumed, along with some knowledge of DDA-type arithmetic. This introduction will serve as a refresher on the processes involved in DA-1 computations.

The DA-1, from the functional standpoint, consists of 108 packages, each containing an integrator and an associated constant multiplier. The integrator receives incremental inputs of two different types, called the dY and dX inputs. The dY inputs are accumulated (= integrated) to form the integrand Y. Increments dX of the variable of integration X control the addition or subtraction of Y into another register called the R register. Overflows of the R register are increments of the integral of Y with respect to X and can be accumulated in another integrator. The integral of Y with respect to X is called Z and the increments of Z are called dZ. The dZ increments can be used as inputs (dX, dY, or both) to other integrators.

The dZ incremental outputs of each integrator also control the addition or subtraction of a constant K into another register called the r register. Overflows of the r register are called the KdZ outputs of the integrator and can serve as inputs (dX, dY or both) to other integrators. They represent increments of the integral of KY with respect to X.

Any integrator or combination of integrators can be made to generate outputs based only on the presence (non-zero) and sign of Y and the presence and sign of dX. In this case $dZ = (\text{Sign } Y)\, dX$ if $Y \neq 0$, $dZ = 0$ if $Y = 0$. This mode of operation is called decision operation and is used for servo operations and certain non-linear computations.

Physically, the registers described above are specific locations on the G-15D drum memory, and combined operation of the two computers is the normal scheme. Besides the registers described above, there are several long lines (address lines) used to specify integrator interconnections and two short lines (Z lines) used to hold integrator outputs until the proper interconnections have been made. Another long line is used to provide scaling and mode control of each integrator.

The DAPPER-1 program converts decimal integrator numbers into bit positions during the specification of integrator interconnections, converts floating point decimal numbers and other scaling data into words and bit positions during numerical input, converts DA-1 words into decimal output form for typeout, and provides the user with other input and control operations. The discussion to follow will indicate the operations which must be performed by more specialized routines of the DAPPER-1 type.

(A.1) -1

II. DA-1 Timing and Control. Drawing 46 c 41 and the DA-1 Functional Diagram are required for this discussion. Copies are enclosed.

A-1 Refer to 46c41.

Near the bottom of the page are shown words 17.06 and 18.06. These contain the integrand and timing control, respectively, for integrator 06. Integrand number I corresponds to word time I all around the drum (00-u7).

From the standpoint of the arithmetic hardware of the DA-1, fixed point arithmetic is used (From the standpoint of problem scaling, floating point arithmetic is much simpler. An explanation is given below).

The integrand Y for integrator I is held in line 17 in a form different from the standard G-15D word. Word 17.I is called the integrand register. Its "fixed" binary point is between T27 and T28. The sign bit for DA-1 arithmetic is in T28; the numerical portion extends from T27 to T1, and is complemented if negative. Incremental inputs to this integrand will be added to the integrand at bit positions controlled by a pulse in M18. These incremental inputs are held in a 4-bit static register (3 bits numerical, one bit for sign) called the dY register, which can hold an accumulated count of from -8 to +7 increments. The accumulated count will be referred to as the ΔY input to distinguish it from a single incremental input dY. The ΔY count is in complementary form if negative.

The complement form of Y and ΔY permits the use of the trailing sign bit (no "second pass" is needed to correct the sign bit as is true of the G-15D AR register).

With the 4 bit ΔY register, it is possible to add the least significant bit of ΔY into the T25 position of the integrand. Other circuitry establishes the low-order (right hand) limit of this addition at T1 (Refer to the description of M18 below).

Since the scaling of ΔY with respect to Y is determined by when the ΔY count is added into the Y register, the relative scaling of dY with respect to Y must be a power of two. In practice, it is quite convenient to use binary Scale Factors for both Y and dY. Experiments with other scaling schemes will indicate the type of difficulties encountered. For the purposes of this discussion, we shall assume binary scale factors on Y and on dY.

Assuming that the maximum zero to full scale magnitude of Y is greater than 1, the floating binary point of Y will lie between 2 bit positions in the integrand. If the scale factor sign convention of DAPPER-1 are employed, then for $2^9 \leq |(Y)_{MAX}| < 2^{10}$, a Scale Factor of + 10 will be used and the binary point will lie between T17 and T18. If the magnitude of a dY increment is 2^{-5} referred to a "one" in the integrand, the dY scale factor is -5, and least significant digit of the Y register should be added to Y in T13 in the example above.

The timing of the Y + dY addition in integrator I is initiated by the right-most bit in M18.I. The bit must be located one bit to the right of the position where the least significant bit of ΔY is to be added in. This bit can be located in any position between T1 and T24 of word I, or it can be located in T29 of word (I-1) (in which case dY is added into T1 of word I). Integrator 00 is an exception in that the start bit must be located between T1 and T24 inclusive of word 00 and cannot be in word u7. See Section B.

If improper scaling is employed, it is possible that when ΔY is added to Y an overflow will occur. Either a positive or negative overflow can occur, and in either case a "one" bit will be written into T29 of word 17.I. The occurrence of this overflow will cause the G-15D overflow Flip-Flop to be set and DA-1 computation will halt immediately. The control aspects of overflows are discussed in Section B. It is advisable during problem preparation to write zeros in T29 of all integrand values to facilitate identification of the faulty scale factor.

The start pulse in word 18.I has been discussed. Two other bit positions (T26 and T28) are of interest.

Position T26 must have a "one" bit. This is a timing control pulse required by the DA-1 hardware and has no programming significance. In effect, the ΔY register is cleared beginning at T27 in preparation for the processing of integrator I +1.

Position T28 specifies whether the output of the integrator is of the dZ=YdX type or of the dZ = (Sign Y) dX type. It is called the "decision" bit. A "one" calls for decision operation. Descriptions of the arithmetic will be found in Section A.3.

To summarize the contents of M18 word I: (1) a start bit must be included, and it can be between the limits of T24 of word I and T29 of word I - 1; (2) a "one" must be placed in T26 of each integrator; (3) Optional bit in T28 calls for decision operation; (4) No other "ones" can be present in M18.

A.2 - 1

A.2 Registers K, R, and r.

The form of the constant K is quite similar to that of Y. The sign bit is in T28; the numerical portion extends from T27 down through T29 of the previous word and is complemented if negative. Since K is not changed by the DA-1, no provision for overflow is needed and T29 is available for numerical use.

Because the KdZ outputs are conditioned on the unmultipled dZ outputs which are not available until T28, the K register for integrator I is found in word time I + 1. Line 15 holds all the K registers.

To anticipate slightly, integrator outputs are the result of accumulated additions of + Y into another register called the R register. The R registers are held in M16 (word I for integrator I). The value of R is restricted to be positive, and bit positions T28 and T29 are not used by the DA-1. R extends from T27 through T1.

The constant multiplier operates like a normal integrator (but with no ΔK inputs) and register r corresponds exactly to register R in function. The r register for integrator I is found in 14.(I + 1). Actually the r register for Integrator I extends from T27 of word I + 1 down through T29 of word I, as does the numerical portion of K. T28 is not used.

A.3 - 1

A.3 Arithmetic for integrator outputs.

The output arithmetic for integrator I will be discussed in 3 parts, YdX outputs, (Sign Y) dX outputs (Decision Operation), and KdZ outputs (from the constant multiplier).

Integration: As T29 of word I-1 is read, the dY register is loaded from one of four dY counters and the DA-1 begins to look for a Start pulse in M18. At the next bit time after the start pulse has been read, two operations begin: (1) ΔY is added to Y; and (2) Y + ΔY is "multiplied" by dX (which can be positive, negative or zero) and added to R. Both of these operations are complete at T28. The R register may or may not have an overflow. (A carry into the T28 position). Lack of an overflow denotes that dZ = 0. Since the R register is restricted to be positive while (Y + ΔY)dX can have either sign, the R register can overflow beyond either the positive or negative limit. The sign of the overflow is the sign of the dZ output (it is also the sign of (Y + ΔY) dX). If this dZ output of integrator I is to be used as an input to another integrator, it must be read during T28 of word I.

Decision: The preceding paragraph assumed that Integrator I was programmed to integrate (i.e., T28=0 in M18). If T28 = 1 in M18, the result of the R + (Y + ΔY) dX addition is ignored. One flipflop has the job of detecting the (non-zero) existence of (Y + ΔY)dX, and at T28 this flipflop is inspected to determine the existence of a dZ = (Sign Y) dX output on decision operation. The output has the sign of (Y + ΔY) dX.

A.3 - 3

Constant Multiplication: The dZ output, existence and sign, is held throughout word time I + 1 to control the addition of KdZ into r. The addition begins at T29 of word I and continues through T27 of word I + 1. The sign of dZ controls complementation of K if necessary. At T28 of word I + 1, the existence and direction of the overflow of the r register provide the existence and sign of the KdZ output of Integrator I. The KdZ output of Integrator I is held until T1 of word I + 2, at which time it is recorded on the two Z lines, as described below.

Note that the dZ outputs are never recorded on the Z lines and that the KdZ outputs are recorded after a delay (from I to I + 2).

A.4 - 1

A.4 Timing of addressing information:

The multiplied output of Integrator I is recorded on two z-lines (M21 for existence, M22 for sign) at T1 of word I + 2. While the DA-1 is in operation, these two lines undergo a precession of 1 bit in each 4 word times, so that the output of Integrator I is in T2 at word I + 6, in T3 at I + 10, and in T27 at word I + 106. Drawing 46 c41 shows the bit positions of the integrator outputs by integrator number during word times 3, 4, 5 (bottom of page) and 6 (top of page).

Because of the precessing scheme based on the 4 word short lines, each integrator output is available for counting only every four word times. Any dY counter, then, must search for four word times in order to be able to accept inputs from each of the 108 integrators, and some means must be provided for indicating to the counter that it has been searching for exactly four word times. As a further consequence of the z-line length, four dY counters are necessary. The most straightforward arrangement of the counters uses the same counter for integrators 00, 04, 08, ..., u0 and u4. A second counter is used for 01, 05, 09, ..., and u5. Similarly with the other two counters. This is suggested and not required, since control of the counter is under control of the programmer as described below. Other arrangements, however, restrict the number of integrator interconnections. In specific problems, this may be feasible and could free long lines for general purpose use.

Each of four dY counters is connected to a long line in the G-15D. Long line 10 is connected to dY counter #4, M11 to #3, M12 to #2, and M13 to #1. Coincidence detection is employed such that if M10 has a "one" bit in some bit position (except T28 or T29), the KdZ output from M21, M22 in the same bit position is algebraically added to the previous contents of the counter. The count is cyclic so that if the counter capacity is exceeded, a large error will occur.

If a "one" is present in T28 of word I (of M10), the dZ output of Integrator I will be counted, (into counter #4) instead of the T28 contents of the Z lines.

The T29 bits of line 10 indicate to counter #4 when to load its accumulated ΔY count into the dY register discussed in Section A.1. If counter #4 is to be used for integrator I, the programmer must place a T29 bit in word I-1 of M10. The occurrence of any T29 in one of the address lines copies the contents of the associated dY counter into the dY register and simultaneously clears the dY counter. If two address lines contain T29 bits in the same word time, the quantity dumped into the dY register will be a bit-by-bit logical sum of the contents of the two counters.

The addressing of dX inputs is identical to that of dY inputs. Lines 6, 7, 8 and 9 are associated with dX counters #4, 3, 2, and 1 respectively. Each dX counter holds only 1 count (existence and sign), and the circuit operation is such that if more than 1 dX input is addressed, each term (existence and sign) will be set as though the multiple counts were presented simultaneously at "OR" gates on the input terms.

The association of specific address lines with specific integrators is at the programmer's discretion. Certain control functions which are fixed onto specific lines are discussed in Section B. The following table indicates the choices made for DAPPER-1 (the DA-1 Addressing Guide is based on DAPPER-1).

TABLE: Memory Assignments for Addressing DAPPER-1

LINE	COUNTER	INTEGRATORS	COUNTER DUMP LOCATIONS IN T29 OF WORDS:
09	dX#1	0 Modulo 4	3 Modulo 4
08	#2	1 " "	0 " "
07	#3	2 " "	1 " "
06	#4	3 " "	2 " "
13	dY#1	0 " "	3 " "
12	#2	1 " "	0 " "
11	#3	2 " "	1 " "
10	#4	3 " "	2 " "

If combined G-15D/DA-1 programs are contemplated, experiments with integrator numbering in a specific problem may indicate that not all counters need to be used for 108 integrator operations. Probably the counters will no longer search in uniform four word blocks. In this event, one of the switches inside the back panel of the DA-1 can be opened, disconnecting the associated counter, and the associated long line can be used for a portion of the general purpose program.

B.1 - 1

B. Control Considerations

B.1 G-15D Commands:

Command 01931 unconditionally starts the DA-1. This command must be operative during word time u7. Artificial + dX inputs are supplied at TO time (=T29 of u7) to integrator 00 regardless of any counter dumps or lack thereof in the dX address lines at TO. Only inputs to integrator 00 are provided and the integrand must be non-zero to obtain outputs.

Command 11931 unconditionally halts the DA-1. Again this command must be operative during word time u7. See the following sections for the other ways to halt the DA-1.

Command 32831 tests the GO flipflop on the DA-1. Test is set if the DA-1 is OFF.

B.2 - 1

B.2 Overflows and Halts:

An overflow in any integrand in the DA-1 immediately halts DA-1 computations and sets the G-15D overflow flipflop.

If dX counter #1 (associated with line 09) contains a non-zero dX at time TO, DA-1 computations will halt (the automatic dX inputs to integrator 00 will no longer be supplied). The G-15D overflow flipflop is not set. In DAPPER-1, dX counter #1 is programmed to work with integrator 00 by means of a T29 in 09.u7. When DAPPER-1 detects a DA-1 halt during normal computation, it next tests the overflow flipflop to decide whether to call for a normal typeout or to call for an overflow-seeking procedure.

B.3 - 1

B.3 Plotters and Plotter/Follower:

The jack marked Plotter on the DA-1 receives the contents of dX counters #3 and #4. Each counter should be set before TO and remain set throughout word 00, although it may be possible in some cases to set the counter in the first few bits of word 00. The address line layout of DAPPER-1 leads to the coding system and interconnection restrictions described in the DA-1 Manual. The rate limitation is that of the plotter, not of the transmitting mechanism.

The jack marked Plotter/Follower on the DA-1, when used for a plotter, receives the contents of dX counter #2 on one axis (during word 00 as described above) and the unmultiplied (dZ) output of integrator u7 on the other (also read during word 00). No dX counter is associated with the output from integrator u7.

When the follower is used, dX signals to the follower are provided by the unmultiplied output of integrator u7 as described above. The ΔY signal from the follower sets dX counter #1 during word 00. If some other input has already arrived at dX counter #1 (or arrives later but before the next Counter Dump signal to the counter), each of the two terms (existence and sign) will be set in "OR" gate fashion.

C.1 - 1

C. Communication between the G-15D and the DA-1:

This section is concerned with the physical characteristics of the information transfer only. The meaning of the transferred information and the programming requirements are discussed elsewhere.

When the DA-1 is OFF, the G-15D has unlimited freedom in reading and writing in all portions of the memory.

When the DA-1 is ON, the G-15D can read any portion of the DA-1 memory without restriction, except that the contents of M21 and M22 will be precessing as described earlier.

Writing into the DA-1 portion of memory has the following limitations:

Line 18 (Control Information). Can be written into in usual fashion. New data must obey the usual instructions regarding the start bit and T26.

Line 17 (Y or Integrand Register). The bit by bit logical sum of the information to be written by the G-15D and the new integrand value ($Y + \Delta Y$) will be written into M17.

Line 16 (R register), Logical sum as above.

Line 15 (K register), Can be written into in usual fashion.

Line 14 (r register), Logical sum as above.

Line 06 - 13, No restriction.

Short lines M21 and M22. Avoid writing in these registers, which are precessing during DA-1 computation. Other considerations involving M21 and M22 are that altering their contents during an interruption in DA-1 computation will make corresponding changes in the first cycle of DA-1 computation when the DA-1 is turned ON again.

Also, the special extract command 32331 was invented for use during DA-1 computation.

ACTION BY: All personnel concerned

REFERENCES: DA-1 Programming and Operations Manual

REMARKS: None

PREPARED BY: R. Keir

APPROVED BY: T. Yamashita

RK/TY/jr

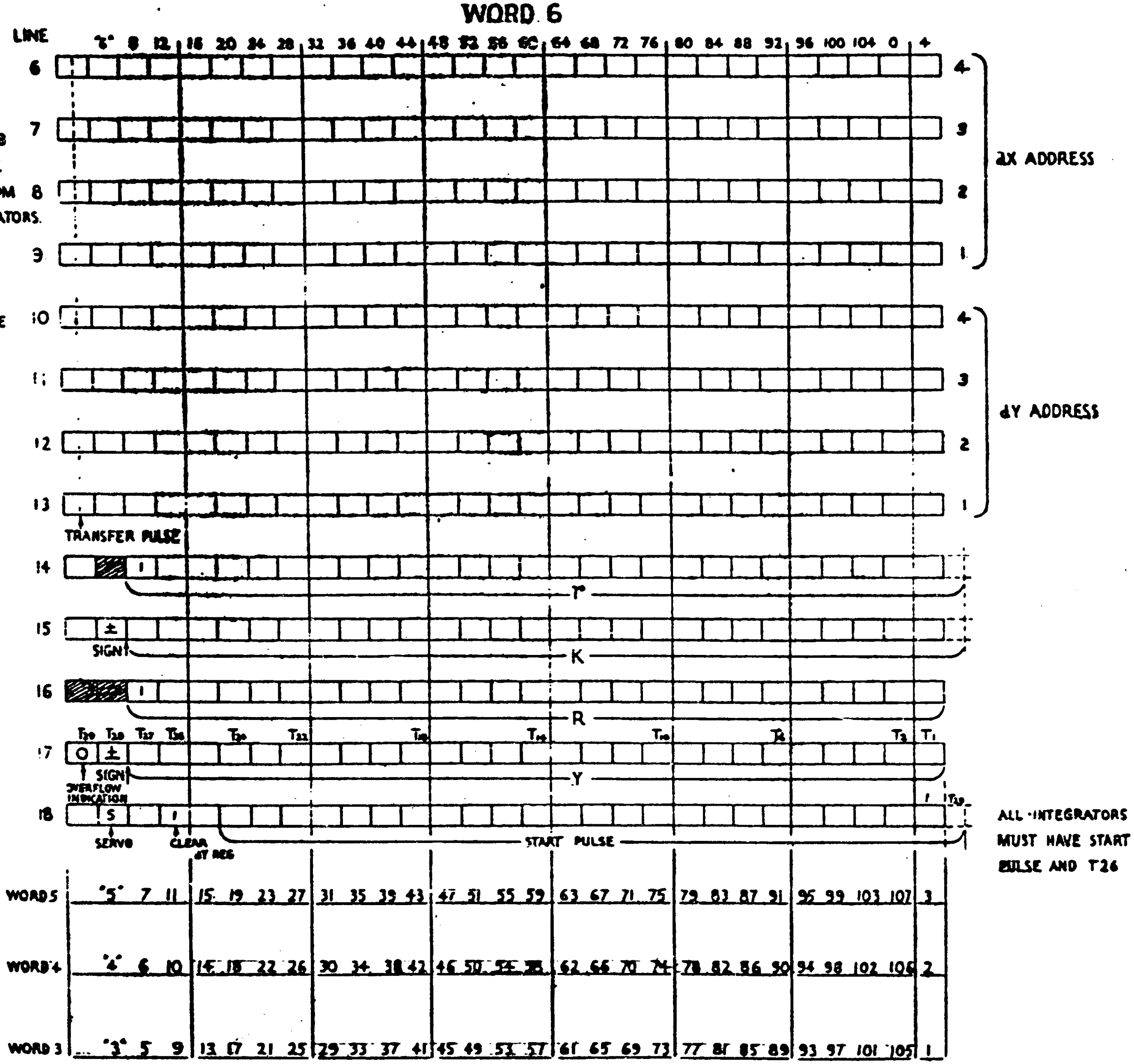
APPENDIX A

Drawing 46 c 41

A	PRODUCTION RELEASE	
B	SEE ECO 604	

WORD 6

INTEGRATOR 7 HAS 4X28=112 OUTPUTS TO DRAW FROM. 108 OF THESE ARE THE K & Z OUTPUTS FROM ALL OF THE INTEGRATORS. THE K & Z OUTPUTS OF 5 AND 6 ARE "OLD". INTEGRATOR 7 ALSO RECEIVES THE dZ OUTPUTS OF 3, 4, 5, & 6.



ALL INTEGRATORS MUST HAVE START PULSE AND T26

INTEGRATOR O MUST NOT HAVE START PULSE IN T29

46C41 B

DA-1			3A3388 3A3393	
NAME	E. R GRANT	R M Beck	R M Beck	
DATE	12-15-56	1-15-57	1-15-57	1-29-57

Bendix Computer

LOS ANGELES, CALIF.

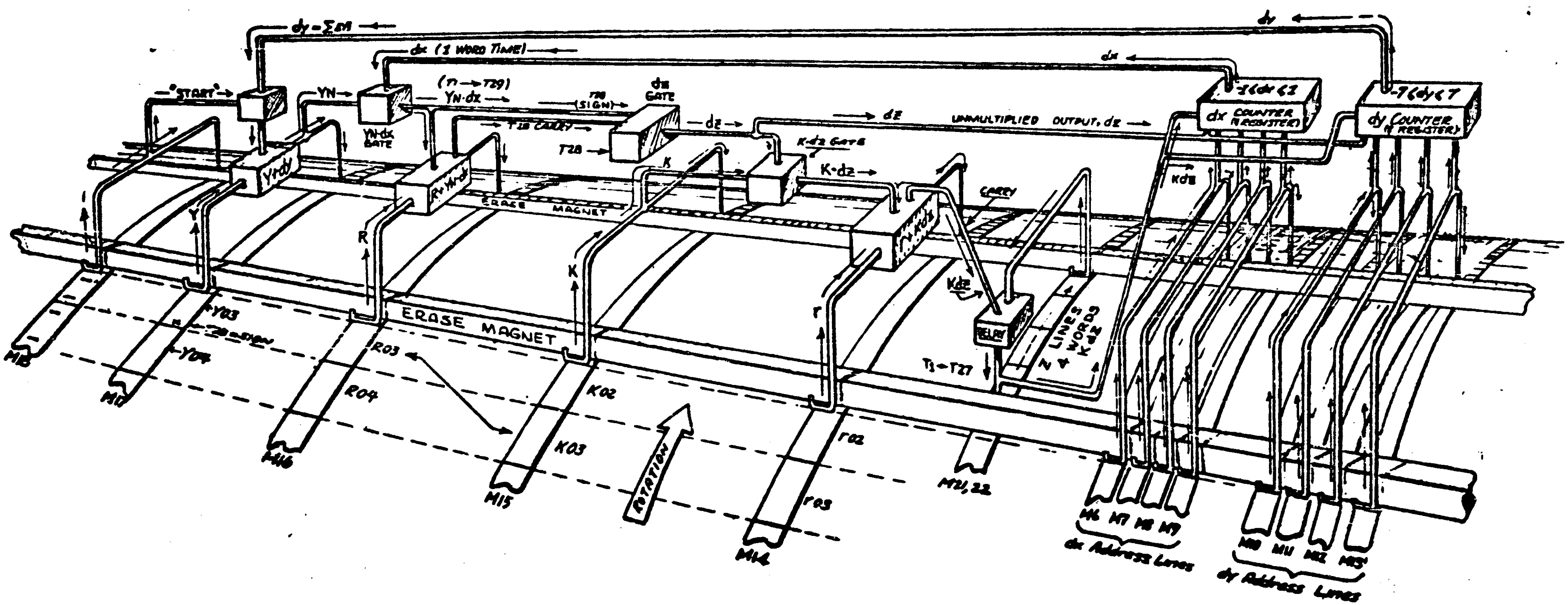
TITLE CHART-PRGM INFORMATION

PART NO. 46C41 B

APPENDIX B

FUNCTIONAL DIAGRAM DA-1

FUNCTIONAL DIAGRAM DA-1



TECHNICAL APPLICATIONS MEMORANDUM NO. 22

25 July 1957

TITLE: Flow Charting Routine

PURPOSE: To provide a neat system of listing the flow of a routine

EQUIPMENT AFFECTED: G-15D

EFFECTIVE DATE: 25 July 1957

INSTRUCTION:

A. Typewriter Set up:

Set the left margin 1" from the left side of the paper. Clear all the tab stops and set a tab stop every 11 spaces from the left margin.

B. Operating Instructions:

1. With Flow Charting Routine on the photo reader, strike <p>.

2. Compute switch to GO. This loads the routine into lines 2, 3, 4, 7, 8, 9, 10, 11 and 0.

3. With the Compute switch OFF, load the routine to be flow charted into line 19. The routine will transfer it to line 18.

4. With the carriage returned, type <scf> and put the Compute Switch to GO.

5. The computer is now ready to accept any one of the following three codes.

a. Type TT - Flow Charting commences at 19.TT. Routine halts at every NCAR, rings bell and waits for a two digit type in for the next command location. Routine halts at all marked transfer commands.

b. Type ± cTT - Same as above except:*

i. If ± cTT is typed and any dummy is sent to destination 28 or 29 and contains a non-zero NN number, this becomes the location of the next command following NCAR.

ii. Complete one line transfer to line c causes a halt.

iii. Mark transfers to line c are treated as subroutine entries and the NN number is saved to be flow charted after all of the main routine is finished. After a space, the main routine is continued at the TT number of the marked transfer command.

c. Type ± LLcTT as in b., except for NCAR. The original routine may have been stored in line LL and executed out of line c. Dummies sent to AR from either line LL or line c are examined for a non-zero NN number.

If at any time during execution of the flow charting routine the operator wishes to specify the next location on NCAR, he throws the compute switch to BP. The routine will halt at NCAR. The operator strikes <f> and throws the compute switch to GO or to BP. A bell will ring and the routine waits for a two digit type in for the next command location.

The routine sends "0" to commands listed except mark and return transfer and NCAR commands. The remainder of line 18 can be typed out as a command and a hex number by typing <sc4f> and GO.

The routine counts carriage returns and after k returns, it carriage returns 6 times, eliminating a type-out of the perforation between sheets. k can be cut in half for double space typing by typing <sc4<u>gf</u>> and GO. The routine asks for type in. Tab "s" sets k for single space typing, and - tab "s" sets for double space typing.

ACTION BY: All personnel concerned

* c = Command Line (6 - 19)

REFERENCES: None

REMARKS: None

PREPARED BY: Dr. H. Huskey and D. Hassell

APPROVED BY: T. Yamashita

DH/TY/bs

EXAMPLE TYPEOUT (REPOSITIONING ROUTINE)

APPENDIX A

```
60        s
                         .00...04.0.29.23
                         .14..u20.0.12.31
                         .20..u20.0.28.31
                         .21...25.0.00.25
                         .55..u58.6.23.24
                         .60..u32.0.24.31
                         .96...97.1.00.25
                         .99..u08.0.24.31
                         .23...27.0.26.00
                         .u2..u05.4.26.30
                         .u3...47.0.26.00
                         .56...57.0.00.25
                         .65..u08.0.24.31
                         .74...83.1.00.25
                         .89..u08.0.24.31
                         .43...45.0.26.28
                         .66...27.3.00.29
                         .78...36.1.28.00
                         .88..u94.0.28.29
                         .67...52.1.28.00
                         .84...91.0.00.21
                         .95..u96.0.01.18
                         .13...15.1.21.28
                         .16...17.1.00.29
                         .19...23.1.28.21
                         .24...63.0.00.29
                         .29...31.0.28.22
                         .33...35.0.21.28
                         .37...39.0.00.29
                         .42...43.0.28.20
                         .44...46.3.00.29
                         .49..u52.0.22.31

      .53...55.0.29.21                          .54...55.0.22.28
      .93..u93.0.28.31                          .64...65.0.31.31..63
      .94..u00.5.21.31                          .63...00.0.01.28
     5.00                                       .92...97.0.28.27

                                  .u0...u1.0.28.28      .u1...03.0.29.23
                                  .13..                 .04...08.4.00.20
                                                        .10...11.0.23.31
                                                        .12...14.0.28.26
                                                        .15..u18.4.28.21
                                                        .18...20.4.31.22
                                                        .22...25.0.22.28
                                                        .26...27.3.00.29
                                                        .28..

                         .28...29.0.28.27

      .30...31.0.00.23                          .31...33.0.22.28
      .32...36.1.00.30                          .34...37.3.20.29
      .41...44.0.22.28                          .38...39.0.28.27
      .45...47.3.00.29
      .48...49.0.28.27            .40...41.0.28.28      .41..
                                  .13..
.50...52.1.00.30        .51..
.59...60.0.26.28
.61..
```

APPENDIX A

.61..u64.4.28.25
.62...63.0.20.28
.64...65.0.31.31..39
.39...00.0.25.18
.68...71.0.21.28
.72..u75.0.23.31
.75...76.3.00.29
.77..

.51...55.0.23.27

.58...12.0.28.28
.13..

.59..

.77...78.0.22.31

.79...81.0.00.30
.75..

.80...76.0.00.29
.82...83.0.28.30
.85..u85.0.28.31
.86..u87.0.29.19
.87...u7.0.26.19
.05...06.0.00.28
.07...03.0.28.02
.11...12.0.09.31
.13..

sc4f

CONSTANTS

.01...03.04.0.29.23	.83043v7
.06..u96.16.0.00.00	.6010000
.08..u00.00.0.31.00	.00003y0
.09..u00.00.0.00.31	.000001z
.17..u01.00.0.00.00	.0100000
.25..u00.32.0.00.00	.00u0000
.27..u00.00.0.00.01	.0000001
.36..u00.00.0.00.03	.0000003
.46...u8.68.0.25.18	.yw44332
.47..u00.00.0.01.00	.0000020
.52..u00.00.0.03.00	.0000060
.57..u00.00.0.05.00	.00000u0
.64..u66.66.0.31.31	.42423zz
.76..u10.00.0.00.00	.0u00000
.81..u16.00.0.00.00	.1000000
.83..u00.00.0.00.16	.0000010
.90..u90.90.0.28.31	.5u5u39z
.91..u01.00.4.00.00	-.0100000
.94..u00.00.5.21.31	-.00006vz
.97..u00.00.0.00.16	.0000010
.98...27.43.0.26.00	.9v2v340
.u6...12.56.4.23.10	-.8w382yu
.u7..u16.03.0.00.00	.1003000

29 August 1957

TECHNICAL APPLICATIONS MEMORANDUM NO. 23

TITLE: Combined G-15D/DA-1 Computations using DAPPER-1.

PURPOSE: To describe procedures for employing G-15D machine language programs with DAPPER-1 for combined G-15D/DA-1 computations.

EQUIPMENT AFFECTED: G-15D and DA-1

EFFECTIVE DATE: 29 August 1957

INSTRUCTION:

A simple modification of the DAPPER-1 program permits automatic exit to a G-15D machine language program at the programmed typeout intervals. The material to follow describes several possible locations for the general purpose program, specifies the exit and re-entry procedure, and suggests several possible systems of control which may prove useful.

GENERAL DISCUSSION

DAPPER-1 was intended as a control routine for DA-1 computations only and is not well adapted to use in combined GP/DDA operation except in the preparation of the program. The amount of flexibility required in a general routine of the DAPPER-1 type is very high and in almost any fixed program it would be possible to provide sufficient output and control capabilities for a given program in a much smaller space. Aside from output conversion, only a very small number of control commands are used during the execution of a program, and these could be incorporated easily in any general purpose program for combined operations.

In planning for combined operations of the G-15D general purpose computer and the DA-1 Digital Differential Analyzer, one must make the decision as to whether or not to use the DAPPER-1 routine during the execution phase of the computation. Probably, it is advisable to prepare the DA-1 portion of the program using DAPPER-1 in either case. Instructions are given below for storing pre-prepared DA-1 programs for use either with DAPPER-1 or with other control routines.

When DAPPER-1 is in place in the G-15, the amount of memory available for General Purpose program is quite small. Therefore, in any program which entails large amounts of General Purpose programming it will be necessary to replace DAPPER-1 entirely. Refer to Technical Applications Memorandum No. 21, dated 15 July 1957, for program requirements.

Where the amount of General Purpose program is small enough to fit into the unused portions of memory, use of DAPPER-1 eliminates the need to write new output routines. The material to follow provides instructions for connecting the GP program to DAPPER-1. The last section describes how to destroy part of DAPPER-1 in order to make room for more GP program.

DUMPING OF MEMORY FOR PROBLEM RE-RUN

This section applies to computations using only the DA-1 as well as to problems involving combined operations. In order to preserve a problem for later insertion into the DA-1 a tape must be prepared which will store relevant information onto paper tape or magnetic tape for later reinsertion. A companion reload routine to this dump program should be written, to be used at the time of reinsertion. The sequence of operation should be as follows:

1. Type in the DA-1 program in the normal fashion.
2. Execute OP code y.
3. Read the memory dump program into M05 and execute it.

The dump program should copy lines 06 through 13 (or as many of the address lines as are needed for the DA-1 problem. See the DA-1 manual) and lines 15, 18 and 17 onto the selected storage medium. The companion reload routine should store the address lines and lines 15 and 18 into their normal locations. It should store the dumped contents of line 17 in either line 05 (for 108 integrator operation) or line 10 (for 81 or less integrator operations). The sequence of operation for reinsertion is as follows:

1. Read in DAPPER-1 in the usual fashion.
2. Read the reload routine into M05.
3. Execute the reload routine. If initial conditions are to be copied into M05, line 23 can be used to copy M19 to M05. Exit to DAPPER-1 can be made by means of a Transfer of Control to 00.00.
4. Use the typewriter to load the typeout list.
5. Use the typewriter to execute OP code 6.
6. Use the typewriter to execute OP code y.

EXIT FROM DAPPER-1 TO THE GENERAL PURPOSE PROGRAM, AND RE-ENTRY

During execution of OP code x+ the DA-1 runs until halted by a dX input to integrator 00, as described in the DA-1 manual. At this time the DAPPER-1 program inspects the typeout list and stores selected integrands in a buffer for later conversion and typeout. Normally as soon as the integrands have been collected DAPPER-1 turns the DA-1 on again and then proceeds with the conversions and typeout. The modification of DAPPER-1 described below will cause control to be transferred from DAPPER-1 to the general purpose program in place of starting the DA-1. The general purpose program must store the contents of the AR register in preparation for re-entry, and it will normally start the DA-1 before re-entry. Conversion and typeout will be performed after re-entry. The enclosed block diagrams indicate the nature of the changes.

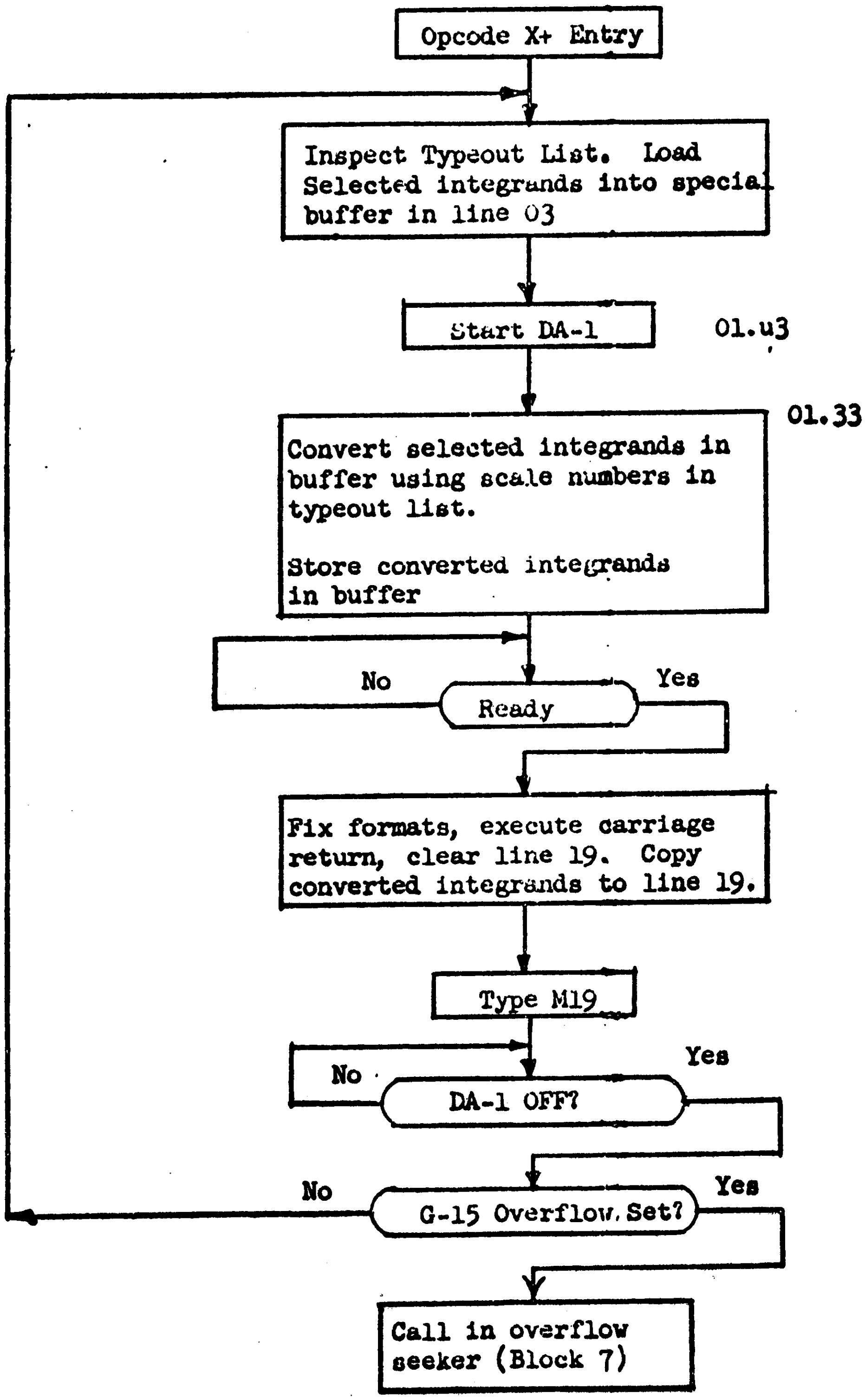
The G-15D portion of the program can start and stop the DA-1 as many times as necessary. The programming instructions are discussed in Technical Applications Memorandum No. 21, which lists the lines which should not be disturbed while the DA-1 is ON.

As an alternative, the General Purpose program can use the normal "typeout" interval to control its own operation. By tallying the number of programmed DA-1 halts, the GP program can decide when to re-enter DAPPER-1 to complete the previous typeout and copy the latest set of integrands into the special buffer in M03.

A loading routine is normally required to read in the general purpose portion of the program before the combined computation is to begin. This loader will substitute a Transfer of Control command for the Start DA-1 Command, which is located in word 01.u3. In the general purpose portion of the program there must be provisions for storing the contents of the AR register upon exit and recopying them into the AR before re-entry. This re-entry is executed by means of a transfer of control to word 01.33 (A Mark Transfer may be used. The Mark will be ignored and destroyed).

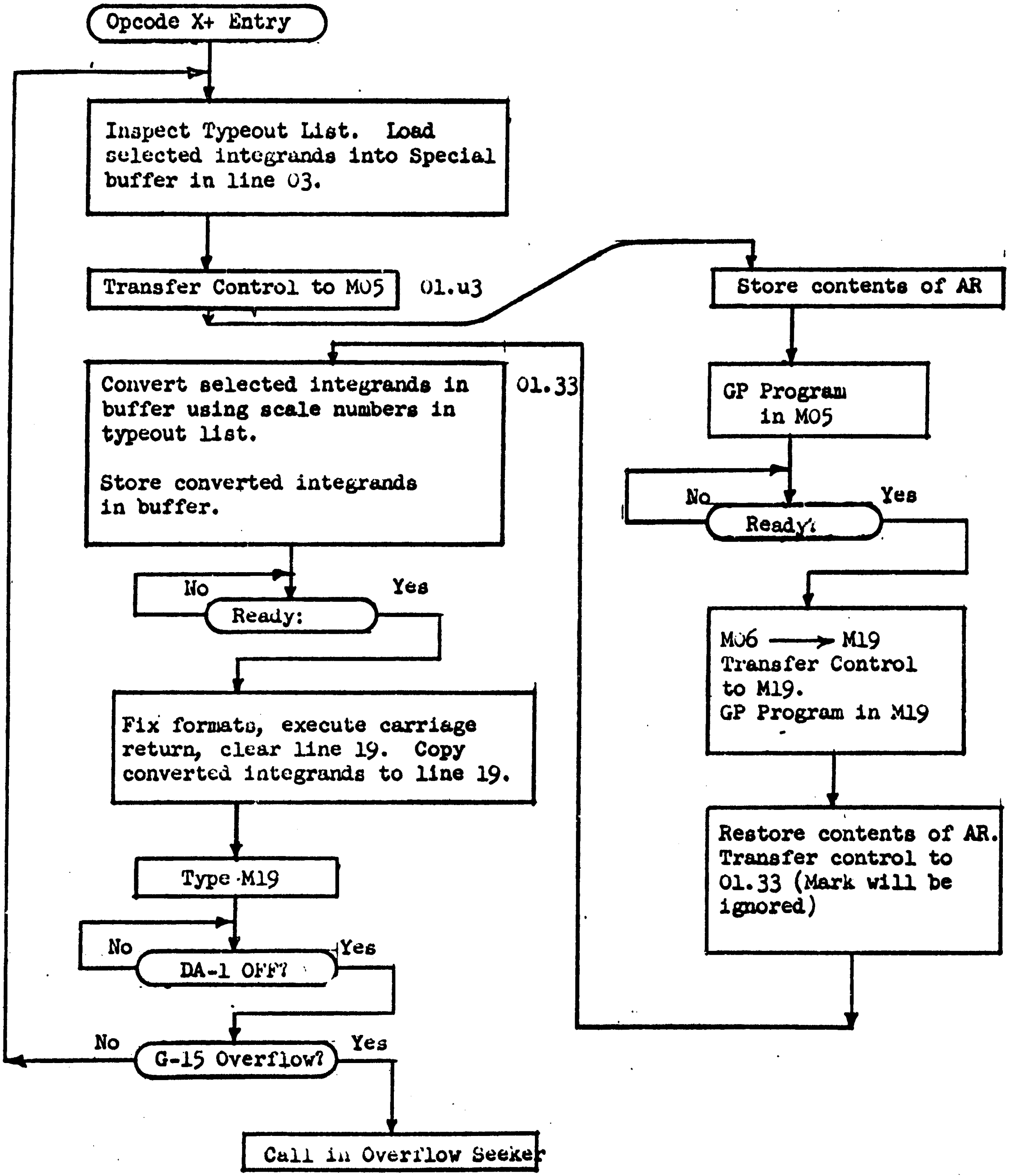
FLOW DIAGRAM

Opcode X+
Unmodified



FLOW DIAGRAM

Opcode X+
Modified for combined G-15D/DA-1 computations



COMBINED OPERATIONS WHEN 81 OR LESS INTEGRATORS ARE TO BE USED

During the readin of the DAPPER-1 program, the operator specifies how many integrators are to be used in the DA-1 portion of the problem. The DAPPER-1 Manual explains which integrators are to be used if 81 or less integrators are required. The manual also indicates which long lines are available for G-15 programs. The operator must remember to flip the proper switches in the DA-1 cabinet to disconnect the input counters.

When 81 or less integrators are to be used, lines 05 and 06 (and perhaps others) can hold G-15D commands. Line 05 is a command line, while the others are not, which suggests that these lines will be copied to M19 for execution. A ready test should be used to prevent copying into M19 while typeout is in progress.

On certain types of problems it is possible that a large number of integrators will be used as constant multipliers. When this is true, all such integrators should be placed in the same modulo 4 group. The associated dY counter can be disconnected and the corresponding dY address line used for GP program. Refer to Technical Applications Memorandum No. 21 for details.

A more restricted class of problems uses a large number of integrators merely to collect answers and no dX inputs are programmed. Grouping these integrators modulo 4 can permit the release of a dX line for GP commands.

COMBINED OPERATIONS WHEN MORE THAN 81 INTEGRATORS ARE TO BE USED

When more than 81 integrators are to be used, no lines are free for GP commands. If the DA-1 program and the GP program have been verified separately, it still may be possible to include a small amount of general purpose programming by destroying the initial condition reset features of DAPPER-1.

The general purpose program should be written to operate from line 05. Exit from and re-entry to DAPPER-1 are as described above. The normal sequence of program preparation is as follows:

1. Enter the DA-1 program, either through the typewriter or from a tape prepared by a memory dump routine. If the typewriter is used, a memory dump at this time would be strategic.

2. Execute OP code y.

3. Load the general purpose program into M05. The loader for this program should modify 01.u3 as described above.

4. Execute the combined program.

No damage is done to DAPPER-1 by this procedure except that OP code y will copy the general purpose program instead of the initial integrand values from M05 into M17. Therefore, when re-starting the combined program, it is necessary to give OP code y (which sets up other registers besides the Y registers), then to read the block of tape containing the initial integrand values into M19 for copying into M17.

USE OF LINE 19 FOR GENERAL PURPOSE PROGRAM

Line 19 can be used to execute general purpose commands which are held on paper or magnetic tape. However, OP code y and all of the output OP codes (u+, v+, and x+) destroy the contents of M19, so that extra care must taken in planning for the use of this line. Specifically, the program in M05 must test ready (to prevent interference with output) and then read in the M19 block of GP program each time the M19 program is to be executed. As soon as this block has been read, it should be rewound (using the tape reverse command) in preparation for later use.

In order for DAPPER-1 to perform its tape search functions properly (to call in the Overflow Seeker, especially) the DAPPER-1 magazine must be left on the photo reader. The tape block containing the general purpose program for M19 can be spliced to the end of the DAPPER-1 tape. Make word 16 of this block (0000008+) or larger. This is a tag number used by the tape search program in DAPPER-1. Tape blocks in DAPPER-1 are numbered sequentially beginning with 0 for the loader.

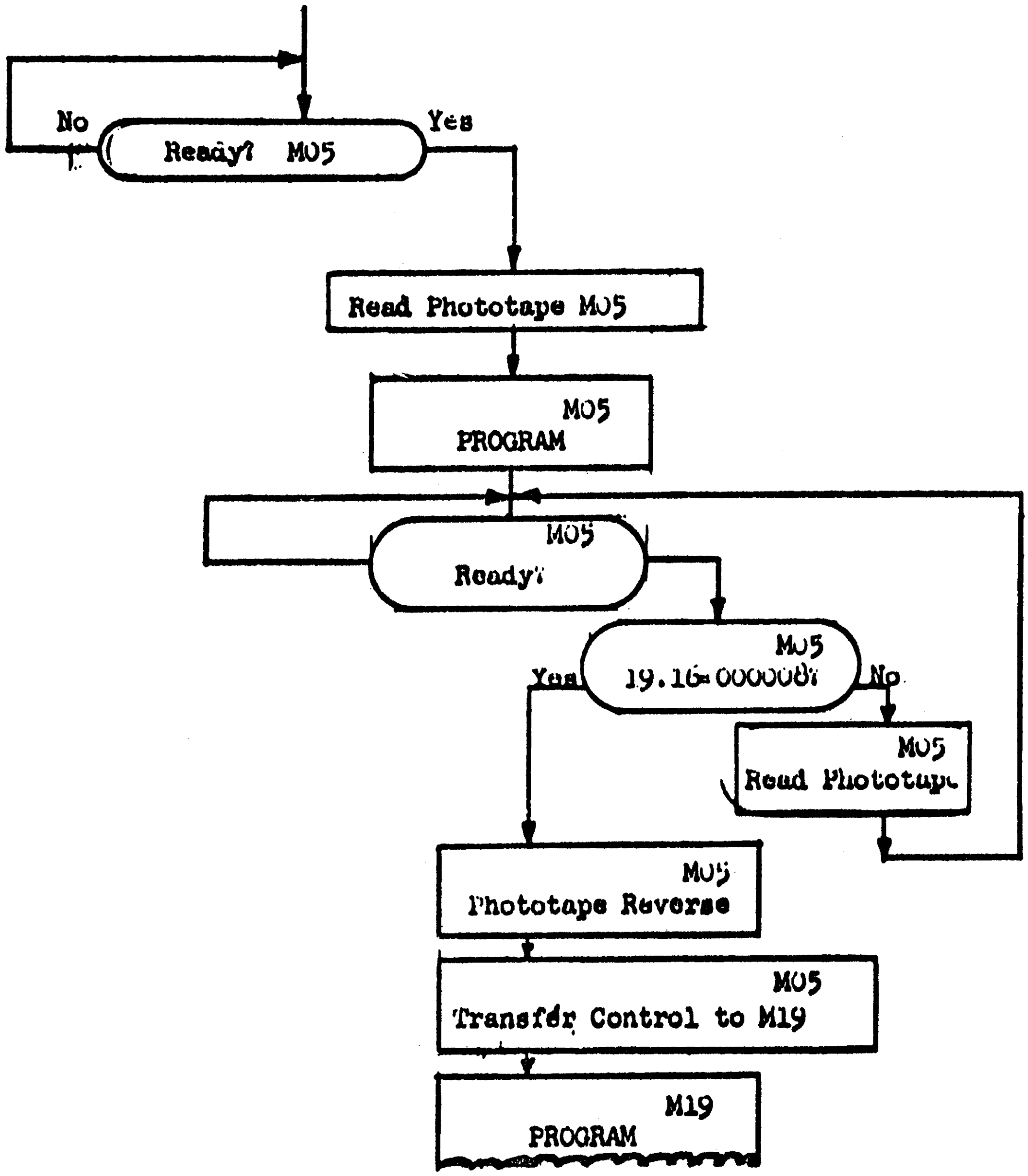
BLOCK NUMBER	OPERATES FROM LINE NUMBER
0 Loader and Memory Setup	M04
1	M00
2	M01
3	M02
4	M03
5 Addressing	
6 Output	M04
7 Overflow Seeker	

The accompanying flow diagram indicates one way of performing the necessary control operations. The test of 19.16 is used to get past the Overflow Seeker at the initiation of computations. A more elaborate test with bidirectional search may be desirable.

BLOCK DIAGRAM - POSSIBLE CONTROL

SYSTEM FOR PROGRAM IN M19

EXIT FROM DAPPER-1



SPACE IN LINE 03 OF DAPPER-1

This procedure is not likely to be as useful as the other suggestions in this memo. The output conversion portion of DAPPER-1 uses so much of the total program space that separation on any large scale is not feasible, and the command locations which could be used for general purpose program during actual computation are fairly few and widely scattered. If the DA-1 scaling has been thoroughly checked so that overflows will not occur, and if most of the input OP codes (including OP code y) can be dispensed with, it is possible to use 78 command locations in M03 of DAPPER-1 for general purpose program. DAPPER-1 no longer controls the movement of paper tape, and automatic overlow detection is no longer provided.

In order to use the available space in DAPPER-1, it will be necessary to prepare a new tape to be inserted in place of one of the blocks of DAPPER-1 after the DA-1 program has been prepared. For safety purposes, it will also be necessary to eliminate those DAPPER-1 OP codes which require the commands which are to be replaced. Normally the sequence of problem preparation will be as follows:

1. Prepare the DA-1 problem using DAPPER-1 as it is.

2. Execute OP code y, which inserts initial integrand values and prepares the DA-1 for computations.

3. Execute OP code u+ which types the initial integrand values. The purpose of executing OP code u+ is to make sure that the output conversion routine is properly loaded into line 04. The tape handling functions of DAPPER-1 are destroyed by the use of the 78 command blocks in line 03 of DAPPER-1.

4. Rewind and remove the DAPPER-1 magazine and replace it with a magazine containing the following tape blocks:

   A. Loader: Copies itself M05 destroying the initial integrand values. Destroys certain OP codes. Loads remainder of general purpose program including the modified block for line 03. Prepares exit from DAPPER-1 to the general purpose program.

   B. Modified version of DAPPER-1, line 03, block 0000004.

   C. Remainder of general purpose program.

5. Read this magazine into the G-15D.

6. Initiate computation using OP code x+.

The following procedure is recommended for the modification of line 03 of DAPPER-1.

1. Using PPR read the 6th tape block of DAPPER-1 into the G-15D. This block can be identified by the contents of word 16. Word 03.16 = 0000004+.

2. Insert general purpose commands in the following locations. 04-09, 11-15, 17-20, 22-56, 58-75, 77-80, 82-87. Do not disturb the contents of the following locations. 00-03, 10, 16, 21, 57, 76, 81, 88-u7.

The loader sets up the exit from DAPPER-1 to the general purpose program by modifying 01.u3 as described above.

OP code entries are in line M00, words 90 through u5. The loader should destroy OP codes 1, 2, 3, 4, 6, 7, w, and y by inserting a ring bell command in each of the appropriate entry words. For convenience the same command (92(90)01731) can be filled in each of the locations 91, 92, 93, 94, 96, 97, u2, and u4. The remaining OP codes (0, 5+, 8+, 9, u+, v+, x+, and z) are unaffected and can be executed by the usual procedure.

ACTION BY: All personnel concerned

REFERENCE: (1) DA-1 Programming Manual

(2) Technical Applications Memorandum No. 21, dated 15 July 1957

REMARKS: None

PREPARED BY: Roy Keir *Roy Keir*

APPROVED BY: T. Yamashita *T. Yamashita*

TECHNICAL APPLICATIONS MEMORANDUM NO. 24A

6 September 1957

TITLE: INTERCOM 1000

PURPOSE: To provide operating instructions and complete information on the use of this interpretive routine.

EQUIPMENT AFFECTED: G-15A and G-15D

EFFECTIVE DATE: 6 September 1957

INSTRUCTION:

A Double Precision Interpretive System has been developed by Bendix Computer Division which will provide answers accurate to 12 decimal digits.

The coding and operating instructions differ from that of Intercom 101 considerably, however, every effort has been made to incorporate as many suggestions as possible which were received from the users and also incorporate those items which were thought to be beneficial in a routine of this type. The basic difference between this interpretive system and Intercom 101 lies in the fact that in one reading the complete program will be in the memory of the machine, therefore, eliminating the need for splicing the program into two sections such as the compiler and interpreter blocks. Some of the advantages of this interpreter system over Intercom 101 are as follows:

1. In the double precision interpretive system all commands that can be obeyed in a program can also be obeyed manually from the computer and conversely. That is, all manual instructions that may be inserted from the keyboard are also possible as program commands in the system. For example, there is a command that converts the whole operating system to a decimal input for individual commands - a single command is typed in and executed - or there is an "automatic" command which converts the whole system into operating at high speed and picking up successive commands from the interpretive memory.

2. Any position in memory can be typed out in sexadecimal form or be typed out in two word groups as a 14 digit floating decimal number.

3. All orders appear in memory as they appear on the programming sheets.

4. Subroutines written in standard machine language may be entered from any position in a program. Return from the subroutine will be to the proper place if the exit command of the subroutines transfers control to line 00 word 47.

5. It is possible to modify any part of a command.

6. All computation can be correct to twelve decimal places.

7. The entire routine is stored in memory at all times.

8. The existence of two mark transfer commands and two return transfer commands makes it possible to have two levels of subroutines besides the main routine.

9. A program can be prepared on an off line flexowriter and read into any line of interpretive memory.

10. The output of floating point numbers with exponents between 40 and 60 can be typed in conventional decimal form with the decimal point placed automatically.

11. There are 1080 interpretive memory positions available excluding subroutines.

12. The list mode provides a choice of the commands to be listed. By setting certain extractors an operator can list commands with any peculiar quality.

13. There are ten B registers available.

14. The fact that all controls are from the keyboard relieves the operator of the burden-some use of the toggle switches.

15. Having double length B registers makes the routine more adaptable to matrix operations. The i and j of a_{ij} can be modified separately and tested for limits separately.

The coding for this system will be in the form of BOPSSTT:

B =	Index Register	(1 digit)
OP =	Operation Code	(2 digits)
SS =	Line Number	(2 digits)
TT =	Word Location in the line.	(2 digits)

(In some instances SSTT will always be 00 depending on the OP code.)

The program utilizes G-15 lines 0 through 8 thus making available 1080 words of interpretive memory, excluding subroutines.

The instructions for coding may take the form ± BOPADDR where:

B =	Index Register	(1 digit)
OP =	Operation Code	(2 digits)
ADDR =	Address	(4 digits)

Then the addressing can be sequential starting from Address 0900, through 1899. By this method a programmer would not be utilizing locations u0 through u7 of lines 9 through 18.

In this system all commands that can be obeyed in the program can also be obeyed manually from the computer and conversely. That is, all manual instructions that may be inserted from the keyboard are also possible as program orders. For example, there is an order that converts the whole operating system to a decimal input for the individual orders to be typed in and executed or there is an "automatic" order which converts the whole system into operating at high speed and picking up successive orders from memory of the computer.

The orders in this system may refer to single precision operations which are done in fixed point binary or they may refer to double precision floating point decimal operations. The result of the single precision operation are stored in what is called the "A" register or single precision accumulator. The results of double precision floating point arithmetic operations are stored in the double precision accumulator or "AA: register.

The primary purpose of the single precision order is to handle B register operations, however: operations with single precision orders are not limited to this. In a general problem using B registers, then it is expected to do the incrementation of the B register and the test of whether they have exceeded the limits or not by use of the single precision orders and by doing binary fixed point arithmetic operations in the A register.

We wish to emphasize that this information is strictly preliminary and that certain functions in the operation codes discussed in this memo is not final until official notice is made. Even though we anticipate very little change we wish to caution all users not to program a large library of routines utilizing the codes contained in this memo.

In the following detailed explanation of orders note that the asterisk is making reference to the fact that the order is single or double precision depending on the sign of the order. All single precision orders are negative and all double precision orders are positive. Orders that do not necessarily refer to single or double precision operations may be either negative or positive. A description of each order follows:

ARITHMETIC OPERATION

Absolute Value *0350000

The order 350000 replaces the number in the "AA" register by its absolute value. A negative number in "AA" is made positive. A positive number in "AA" is unchanged.

Example:

The "AA" register contains the number -51.123456789645. After the order 0350000 is obeyed, the "AA" register contains the number 51.123456789645.

Clear and Subtract *B40SSTT

The order 040SSTT replaces the contents of the "AA" register by the contents of addresses SSTT and SSTT+1, with reverse sign. TT must be even.

Example:

Addresses 10.14 and 10.15 contain the number 51.123456789654. After the order 0401014 is obeyed, "AA" contains-51.123456789654. The contents of 10.14 and 10.15 are unchanged.

Subtract *B41SSTT

The order 041SSTT subtract the contents of addresses SSTT and SSTT+1 from the contents of the "AA" register. The difference replaces the previous contents of the "AA" register. TT must be even.

Example:

Addresses 10.14 and 10.15 contain the number 51.123456789655 and the "AA" register contains the number 51.654987654321. After the order 411014 is obeyed, the "AA" register contains the number 51.431530764667.

Clear and Add *B42SSTT

The order 042SSTT replaces the contents of the "AA" register by the contents of address SSTT and SSTT+1. TT must be even.

ARITHMETIC OPERATION (Continued)

Clear and Add (Continued) *B42SSTT

Example:

Addresses 10.14 and 10.15 contain the number 51.123456789654. After the order 0421014 is obeyed, "AA" contains 51.123456789654. The contents of 10.14 and 10.15 are unchanged.

Add *B43SSTT

The order 043SSTT adds the contents of addresses SSTT and SSTT+1 to the contents of the "AA" register. The sum replaces the previous contents of the "AA" register. TT must be even.

Example:

Addresses 10.14 and 10.15 contains the number 51.123456789654 and the "AA" register contains the number 51.465987654321. After the order 431014 is obeyed, the "AA" register contains the number 51.589444443975.

Multiply B44SSTT

The order 044SSTT multiplies the contents of the "AA" register by the contents of addresses SSTT and SSTT+1. The product replaces the previous contents of the "AA" register.

Example:

Addresses 10.14 and 10.15 contain the number 51.200000000000. The "AA" register contains the number 51.300000000000. After the order 0441014 is obeyed the "AA" register contains the number 51.600000000000.

Divide B48SSTT

The order 048SSTT divides the contents of the "AA" register by the contents of address SSTT and SSTT+1. The quotient replaces the previous contents of the "AA" register.

Example:

Addresses 10.14 and 10.15 contains the number 51.400000000000 and the "AA" register contains the number 51.800000000000. After the order 0481014 is obeyed, the "AA" register contains the number 51.200000000000.

ARITHMETIC OPERATION (Continued)

Store *B49SSTT

The effect of the order 049SSTT is to store the contents of the "AA" in locations SSTT and SSTT+1 replacing the contents of SSTT and SSTT+1.

Example:

Memory locations 10.14 and 10.15 contain the number 51.123456789465 and the "AA" register contains the number 52.987654321456. After the order 0491014 is obeyed, locations 10.14 and 10.15 contain the number 52.987654321456.

SINGLE PRECISION FIXED POINT ARITHMETIC

Absolute Value *-0350000

The order -0350000 replaces the number in the "A" register by its absolute value. A negative number in "A" is made positive. A positive number in "A" is unchanged.

Example:

The "A" register contains the number -1234567. After the order 0350000 is obeyed, the "A" register contains the number 1234567.

Clear and Subtract *-B40SSTT

The order -04SSTT replaces the contents of the "A" register by the contents of address SSTT with reversed sign.

Example:

Address 10.14 contains the number -2345678. After the order -0401014 is obeyed "A" contains 2345678. The contents of 10.14 are unchanged.

Subtract *-B41SSTT

The order -041SSTT subtracts the contents of address SSTT from the contents of the "A" register. The difference replaces the previous contents of the "A" register.

SINGLE PRECISION FIXED POINT ARITHMETIC (Continued)

Subtract (Continued) *-B41SSTT

Example:

Address 10.14 contains the number 1234567 and the "A" register contains the number 8888888. After the order 0411014 is obeyed, the "A" register contains the number 7654321.

Clear and Add *-B42SSTT

The order -042SSTT replaces the contents of the "A" register by the contents of address SSTT.

Example:

Address 10.14 contains the number 01000000 After the order 04210.14 is obeyed, "A". contains 01000000. The contents of 10.14 are unchanged.

Add *-B43SSTT

The order -C43SSTT adds the contents of address SSTT to the contents of the "A" register. The sum replaces the contents of the "A" register.

Example:

Address 10.14 contains the number 1234567 and the "A" register contains the number 1111111. After the order 0431014 is obeyed the "A" register contains the number 2345678.

Store *-B49SSTT

The effect of the order -049SSTT is to store the contents of "A" in location SSTT, replacing the contents of SSTT.

Example:

Memory location 10.14 contains the number 1234567 and the "A" register contains the number 7432156. After the order 0491014 is obeyed, location 10.14 contains the number 7432156.

TRANSFER OF CONTROL OPERATIONS

Orders are normally obeyed in sequence. After an order located in position SS.TT has been executed, the next order is taken from position SS.TT + 1. The following operation codes make it possible to interrupt this normal flow; that is, to transfer control to an order not necessarily immediately subsequent. In some instances this transfer of control is conditional upon the sign of the "AA" or "A" register, in other instances it is not. These latter operations cause control to be transferred to the order located in the address portion of the transfer of control order itself, regardless of what the contents of any particular register may be.

Unconditional Transfer B36SSTT

The order 036SSTT transfers control to the order located in position SS.TT.

Non Negative Transfer *B20SSTT

The order 020SSTT transfers control to SSTT if the contents of the "AA" register is non-negative. If the contents of the "AA" register are negative, the next order will be taken in sequence.

Negative Transfer *B22SSTT

The order 022SSTT transfers control to SSTT if the sign of the "AA" register is negative. If the contents of the "AA" register are non-negative (≥ 0) the next order will be taken in sequence.

Marked Transfer

There are two marked transfer orders which stores the address of the location in which they are stored and transfers control to a new position in memory. Corresponding to these two orders are two return orders which will return to the respective location specified by the last marked transfer command. This makes it possible to have two levels of subroutine operation as well as operation at the main level; that is, a first marked transfer can provide for a transfer of control to a subroutine and while in that subroutine the second marked transfer can transfer control to another subroutine.

Marked Transfer "One" B26SSTT

The "Marked Transfer One" order is identical to the unconditional transfer operation, except that the location number is stored in a special register not addressable by the programmer. The purpose of the storage is made clear by the "return transfer one" order.

TRANSFER OF CONTROL OPERATIONS (Continued)

Marked Transfer "One" (Continued) B26SSTT

Example:

The order 0261877 is stored in location 11.13. Upon execution of this order, the address 11.13 is stored in a special register and the next order is obeyed from 18.77.

Marked Transfer "Two" B28SSTT

Operation of this order is identical to Mark Transfer "One".

Return Transfer "One" 0160000

The order 160000 transfers control to the order located immediately subsequent to the last Marked Transfer "One". This order is meaningless unless preceded sometime in the program by a Marked Transfer "One" order.

Example:

If a sequence of orders (not including a marked transfer "one") is obeyed followed above example, the order 160000 returns control to the order located in address 11.14.

Return Transfer "Two" 0180000

The order 0180000 transfers control to the order located immediately subsequent to the last Marked Transfer "Two". This order is meaningless unless preceded sometime in the program by a Marked Transfer "Two" order.

Exit to a Subroutine B08SSTT

Subroutines may be written in machine language and stored in lines 09, 10 through 18. Entry to these subroutines can be made by giving the command 008SSTT where SS equals the line in which it is stored and TT equals the entry into the subroutine.

Example:

Find the square root of x. x is stored in 09.14-15. The square root subroutine is stored in line 14 and the entry to the square root routine is u5. Two orders will have to be given as follows:

1.	0420914	clear and add x into "AA"
2.	00814u5	exit to subroutine

The routine will exit to the subroutine, get the $\sqrt{x}$ and return to the desired place in a routine, if in automatic, or to manual, if in manual, with the square root of x in "AA".

INPUT OPERATIONS

Type in Decimal Numbers B24SSTT

The computer responds to the order 024SSTT by typing SSTT followed by a tab. This indicates that the computer is prepared to accept a 14 digit floating point number in locations SS.TT and SSTT+1. TT must be even.

The operator now types his floating point decimal number in the format: dd.dddddddddddd sign followed by tab "s".

The computer converts this number and places it in the interpretive memory. It then returns the carriage and types SSTT+2. Again the computer is ready to accept either another number followed by a tab "s" or an order followed by tab, slash "s". If an order is typed followed by a tab, slash "s" the computer will execute the order and return to the manual input state. (See page 13, Control Operations, Manual).

Type in Orders or Single Precision Numbers -B45SSTT

The computer responds to the order -045SSTT by typing SSTT followed by a tab. This indicates that the computer is prepared to accept information in the designated location.

The operator then types an order or a single precision number followed by tab "s".

This input goes through no conversion routine, therefore the exact "type in" is stored in SSTT.

After the computer stores the input, a carriage return is executed and the next location number is typed in the left hand margin. The computer is again ready to accept input.

If it is desired to change modes of operation, type an order, a tab and the slash "s". The routine will execute the order typed immediately before the slash. The ring of the bell signals the return to the manual input state.

Read Paper Tape B78SS00

The order 078SS00 initiates the process of reading a punched tape into consecutive locations of line SS starting at word 00. If in the "automatic" mode, computation can continue until an input or output order is reached.

OUTPUT OPERATIONS

Type and Tab *B32SSTT

The order 032SSTT causes the contents of SSTT and SSTT+1 to be typed in floating decimal form, followed by a tab. TT must be even.

Type and Carriage Return *B34SSTT

The order 034SSTT causes the contents of SSTT+SSTT+1 to be typed in floating decimal form, followed by a carriage return. TT must be even.

Type Fixed Point B33SSTT

The order 033SSTT results in the type out of a floating decimal number located in SSTT and SSTT+1 in fixed point notation with the decimal point automatically positioned. A number whose exponent is not between 40 and 60 will be typed, but the type out will be of no significance. If it is anticipated that a routine will generate answers whose floating point exponents are not within this range, it is advised that the routine be written for type outs in floating point. This can be accomplished by using one of the orders B34SSTT or B32SSTT which are explained above.

Fix Format B30SSTT

The order 030SSTT provides for the automatic positioning of the typewriter carriage by the execution of a specified number of carriage returns followed by a specified number of tabs. The address digits of this order are a code indicating the number of tabs and carriage returns to be executed. TT will equal the number of CR and SS equals the number of tabs. TT should never exceed 31.

Memory to Paper Tape B79SS00

Any line of memory may be punched on paper tape upon execution of the order 079SS00 where SS designates the line to be punched. When the order is executed line SS is transferred to line 19 and punching begins. If in the "automatic" mode computation will continue until another input or output is reached.

CONTROL OPERATIONS

Manual -0570000

Upon execution of -0570000* a mode is set so that one order at a time can be typed and executed. Every routine should have this for the ending command.

Automatic -059SSTT

Upon execution of the order -059SSTT, computation will begin at high speed beginning with location SSTT. The computer will continue to execute orders in this mode until an order is obeyed that transfers control to "manual" operation.

If the operator desires to break into the routine at any time, he should place the compute switch in breakpoint position. When the machine halts, with compute off type the "f" with enable on. This will return the system to a mode in which one order can be typed in and executed. Upon execution of this one order, the system will automatically return to the point in the routine where it left off and continue operation from there.

Set Selective Print -0290000

The order -0290000 sets a mode that provides for certain type outs upon execution of the "Automatic" command. The type outs will occur when control is transferred to automatic. The type outs are the location, and order and the result of the execution of this order. These type outs are determined by the contents of memory positions 08.82 and 08.83. In this mode a portion of each order is extracted as determined by an extractor (contents of 08.83) and is compared to a code (contents of 08.82), and if these agree then the appropriate order is to be listed. This is accomplished by setting a program switch which says "set list". If the extract portion does not agree with the code then the order will be executed without any typing. This makes it possible to list any order in a routine.

	Extractor	Code
Line	000zz00	000LL00
Word	0zz00zz	0SS00TT
OP Code	0zz0000	0"OP"000
B Register	z000000	B000000

* When an order requires no specific reference to an address in the operating memory, 0000 is written in the address portion.

CONTROL OPERATIONS (Continued)

Set Selective Print (Continued)

Example:

To type out all orders with OP Code = 49 store 0zz0000 in 08.83 and 0490000 in 08.82, give the order -0290000, and the order -0591023. Computation will begin at location 10.23 and list all orders with OP = 49.

Remove Selective Print -0310000

The order -0310000 removes the list mode so that computation will continue at high speed.

G-15D OPERATING INSTRUCTIONS

I Loading Procedure

1. Place the Intercom 1000 magazine on the photo reader in the rewind position. The first block is the loader.

2. Strike the "p" key with enable switch on.

3. When reading stops place the compute switch in the "GO" position.

4. The photo reader will read the entire program. When reading is completed a bell will ring with the input-output lights in the slow in configuration (input-output lights) or 00000. The program is in MANUAL. Compute switch should be left in the "GO" position.

5. At this point 1080 interpretive memory locations are available; excluding subroutines.

6. If subroutines are required they can be read into the memory at any desired location. (See Section III)

II Operating Modes

There are three operating modes when using this routine:

1. Manual

2. Manual type-in

3. Automatic

The manual mode can be selected at any time by striking the "f" key with enable switch "ON" and placing the compute switch back to "GO".

The manual type-in mode can be selected by inserting the instruction for typing in commands or numbers if the machine is in the mode 1.

The automatic mode is selected by an instruction when a program is to be executed automatically.

Occasionally it is desired to have the computer execute particular instructions rather than to store the instructions in memory. If the instructions should be obeyed then the instruction should be typed, followed by a "tab /s" rather than typing the instruction and "tab s".

When commands are being typed, preceeding zero's do not have to be entered.

NOTE: By use of the above modes of operation each instruction can be executed independently or can be stored in memory for automatic computation.

G-15A OPERATING INSTRUCTIONS

IA Loading Procedure

1. Place the Intercom 1000 magazine on the photo reader in the rewind position. The first block is the loader.

2. Strike the "p" key with enable switch on.

3. When reading stops strike the "f" key with enable switch on.

4. Move the command line switch to line 19.

5. Move the compute switch to B.P.

6. Move the compute switch to idle.

7. Move the command line switch to center position.

8. Move the compute switch to "GO".

9. The photo reader will read the entire program. When reading is completed a bell will ring with the input-output lights in the slow in configuration (input-output lights) or 00000. The program is in MANUAL. Compute switch should be left in the "GO" position.

10. At this point 1080 interpretive memory locations are available; excluding subroutines.

11. If subroutines are required they can be read into the memory at any desired location. (See Section III)

IIA Operating Modes

There are three operating modes when using this routine:

1. Manual

2. Manual type-in

3. Automatic

The manual mode can be selected at any time by striking the "sf" key with enable switch "ON" and placing the compute switch back to "GO".

The manual type-in mode can be selected by inserting the instruction for typing in commands or numbers if the machine is in the mode 1.

The automatic mode is selected by an instruction when a program is to be executed automatically.

Occasionally it is desired to have the computer execute particular instructions rather than to store the instructions in memory. If the instructions should be obeyed then place the compute switch "OFF", put the enable switch on and strike "sf" and then move the compute switch to "GO" and type in the instruction followed by "tab s".

III Use of Subroutines

Various subroutines are available for use in Intercom 1000. In general, the following information will allow one to utilize those subroutines.

1. Storage. A subroutine may be stored in any part of the interpretive memory (lines 09 through 18) at the option of the programmer.

2. Exit to Subroutine. The following programming steps are required upon exiting to a subroutine:

   a. The argument is placed in the AA register (address 21.00).

   b. The command effecting the exit to the subroutine is then given. It is of the form:

      B 08 ADDR

      The first two digits of the address indicates the line in which the subroutine is stored, and the last two digits the entry location of the subroutine. (See list below of entry locations of subroutines which have already been completed).

   c. The output of the subroutine, that is, the result of the operation performed by the subroutine, will be in the AA register.

Subroutine entry locations:

a. Square Root u5

b. Sine-cosine

   Sine (radians). 42

   Sine (degrees). 39

   Cosine (radians). 26

   Cosine (degrees). 23

c. Exponential functions

   10^x 72

   e^x 22

   2^x 08

d. Logarithms

   $Log_{10}x$. 71

   $Log_e x$ 17

   $Log_2 x$ 08

IV B Register Operations

The function of a B register is to atuomatically modify the address portion of a command. Any time a B register is named in a command the address portion of that command will be modified by the contents of the B register specified. This effects no permanent change to the command itself but has the result of modifying this command during its execution.

All B registers are made up of two words as is shown on Appendix I. It should be understood that both words are added to a command any time a B register is named in a command, therefore, it is important for the programmer to know the contents of both sides at all times.

Having each B register occupy two words gives the programmer the opportunity to modify either SS or TT of a command and test either against a limit.

Constants for B register settings are not stored in their decimal form. The following simple procedures outlines the method necessary to arrive at the desired setting.

If it is desired to set B register "4" equal to N for modification of the SS portion of the address proceed as follows:

$$\frac{2N}{16} = Q + \frac{R}{16}$$

The desired setting to be stored in either 07.52 or 07.53 is "0000QR0". If either word of a B register is not used be sure that it is set to zero before the other side is used. A suggestion for keeping up with these seetings is to always set TT in the even word of the two word register and SS in the odd word of the B Register.

If it is desired to set a B register equal to N for modification of the TT portion of a command proceed as follows:

$$\frac{N}{16} = Q + \frac{R}{16}$$

The desired setting to be stored in 07.50 or 07.51 is "QR00000" where R may equal 1 through 9 or u, v, w, x, y, z (v, v, w, x, y, z equals 10, 11, 12, 13, 14, 15 respectively) and $0 \leq Q \leq 6$. It will never be necessary for Q to be larger than 6 since TT can never to incremented by more than 107.

To assist one to follow the use of this routine Technical Application Memorandum, Evaluation of Polynomial is attached as a guide.

SUMMARY

ARITHMETIC OPERATIONS

*35 Absolute Value

*40 Clear and Subtract

*41 Subtract

*42 Clear and Add

*43 Add

44 Multiply

48 Divide

*49 Store

TRANSFER OPERATIONS

36 Transfer Control

*20 Transfer Positive

*22 Transfer Negative

26 Mark Transfer (1)

28 Mark Transfer (2)

16 Return Transfer (1)

18 Return Transfer (2)

08 Exit to Subroutine

INPUT OPERATIONS

24 Type in Double Prec. Numbers

45- Type in Commands

78 Read Punched Tape

OUTPUT OPERATIONS

*32 Type and Tab

33 Type Fixed Point

*34 Type and Carriage Return

30 Fix Format

79 Punch Tape

ADDRESSABLE MEMORY

Interpretive Memory Line 9 through 18

Double Precision Accumulator "AA" 21.00-01

Single Precision Accumulator "A" 22.03

B Registers 07.46-65

CONTROL OPERATIONS

57- Manual

59- Automatic

29- Set Selective Print

31- Remove Selective Print

* Single or Double Precision

All Single Precision Operations are initiated by the execution of negative commands; all Double Precision Operations are initiated by the execution of positive commands.

EXAMPLE FOR INTERCOM 1000 For G-15D

1. p key with enable switch on, photo reader reads first block (loader)

2. Compute switch to GØ, program will be read into memory. When reading is completed a bell will ring. Leave the compute switch ON.

3. 450900-tab s (Command to set up manual type in of commands)

```
 9.00  421300  s      ┐
 9.01  431300  s      │
 9.02  441500  s      │
 9.03  342100  s      ├  See explanations for commands
 9.04  332100  s      │
 9.05  570000-  s     ┘
 9.06  241300  /s        To type in numbers tab /s is required so that this instruction
13.00  53.100000000000  s         will not be stored. If the instruction is to
13.02  241500  /s                 be stored then tab s. etc.
15.00  52.100000000000  s
15.02  590900-  /s
```

54.200000000000
2000.00000000

EXAMPLE FOR INTERCOM 1000 For G-15A

1. p key with enable switch on, photp reader reads first block (loader)

2. f key with enable on, command switch to L19, compute switch to BP, command switch center, Compute switch to GØ. When reading is completed program in manual. Leave compute switch ON.

3. 0450900-tab s (Command to set up manual type in of commands)

```
 9.00  0421300  s
 9.01  0431300  s
 9.02  0441500  s
 9.03  8342100  s
 9.04  0332100  s
 9.05  0570000-  s
 9.06  Compute Switch off, enable switch on type "sf", compute switch to GO. Program will
           return to manual.

       0241300  s
13.00  53.100000000000  s
13.02  sf  see above sequence   0241500  s
15.00  52.100000000000  s
15.02  sf  see above sequence   0590900-  s
```

54.200000000000
2000.00000000

CHECK SUMS

G-15D

.48w173x	Loader
-.6v69744	Line 2
-.xx0w9x4	Line 3
.w606581	Line 4
-.uwz85ww	Line 5
.9141y03	Line 6
-.6y73350	Line 7
.6z86647	Line 8
-.2yxz6wv	Line 0
-.3zw8z26	Square Root
.87z9x80	Sine, Cosine
-.6w70532	e^x, 10^x, 2^x
-.u46uyzu	$Log_e x$, $Log_{10} x$ $Log_2 x$

G-15A

.3045z52	Loader
.23yw593	Line 0
-.4889xu2	Line 2
-.wxz1415	Line 3
.w606421	Line 4
-.y8z78uy	Line 5
.9141y03	Line 6
-.58x12v0	Line 7
-.73v0w7y	Line 8
-.1199ww8	Square Root
-.0120544	Sine, Cosine
.18x22x2	e^x, 10^x, 2^x
.863x029	$Log_e x$, $Log_{10} x$, $Log_2 x$

REFERENCES: Technical Applications Memorandum No. 24

REMARKS: Note that the G-15A and G-15D version of this routine are identical, except for the operating instructions which are on page 14A.

Tapes punched by Intercom 1000 for the G-15D and G-15A are interchangable.

PREPARED BY: Bendix Computer Division

APPROVED BY: T. Yamashita

9-6-57
bl

B REGISTERS	LOCATIONS
1	07.46-47
2	07.48-49
3	07.50-51
4	07.52-53
5	07.54-55
6	07.56-57
7	07.58-59
8	07.60-61
9	07.62-63
u	07.64-65

Furnished with the INTERCOM 1000 tape are a number of subroutines written in machine language. It is realized that these subroutines will not satisfy all the needs of the users. The following information will be of value to users who have sufficient need to warrant coding of machine language subroutines to be used with INTERCOM 1000.

I. Floating Point Arithmetic Subroutines within INTERCOM 1000

			Word Position
A.	Storage - Line Four		
B.	Execution - Line One		
C.	Entry -	Clear and Add	42
		Add	43
		Clear and Subtract	40
		Subtract	41
		Multiply	44
		Divide	48
		Absolute Value	35
D.	Accumulator - 21.00-01		
E.	Operand - $ID_{0,1}$		
F.	Exit - Line 4, Word 50		

II. All "user" prepared subroutines should be executed from Line One.

III. Upon exit from a subroutine store 34342vz (G-15D) or 343429z (G-15A) in Line Four, Word 50 and finally transfer control to Line 0, Word 47.

SUMMARY:

To use one of the floating decimal subroutines built into INTERCOM 1000, use 21.00 - 01 for the accumulator and $ID_{0,1}$ for the location of the operand. Store an exit in line 4, word 50 and transfer line 4 into line one for execution. The exit from the subroutine should transfer the "user" prepared subroutine into line one where execution will continue.

The "user" prepared subroutines should exit by storing 34342vz (G-15D) or 343429z (G-15A) in word 50 of line 4 before finally transferring control to line 0 word 47.

SUPPLEMENT NO. 1

TECHNICAL APPLICATIONS MEMORANDUM NO. 24A

1 November 1957

TITLE: INTERCOM 1000

PURPOSE:

Part I	Automatic "B" Register
Part II	Card Input-Output Routine
Part III	Flexowriter Input Routine
Part IV	Tape Format Modifier
Part V	Revisions to Intercom 1000
Part VI	Debugging Procedure for Intercom 1000
Part VII	Intercom 1000D - Check Sums

EQUIPMENT AFFECTED: G-15D

EFFECTIVE DATE: 1 November 1957

INSTRUCTION:

PART I

AUTOMATIC INDEX REGISTER SUBROUTINES

Instructions: With the addition of the Automatic Index Register Subroutine, INTERCOM 1000 (G-15D) includes twenty sets of index registers which provide a simple means of modifying the address portion of any order before it is obeyed. This enables certain orders in a sequence to operate on the contents of different memory locations each time the sequence is repeated. Furthermore, these index registers provide a means for the transfer of control after a sequence of orders has been repeated a specified number of times. The three registers in each set "k" are called B(k), D(k), and L(k) where k is an integer in the range from 1 through 9 and u(10) designates the set to which they belong. Numbers in the index registers are not in floating point form.

Ten sets of index registers are used to modify the last two digits (TT) of the address portion of a command. In the discussion to follow, these index registers will carry the subscript (T), i.e., B_T, D_T, and L_T. The four commands that are used to modify the TT portion of a command are:

k	OP	ADDRESS	
k	70	00TT	Set B_T register
k	71	00TT	Set D_T register
k	72	00TT	Set L_T register
k	76	SSTT	Increment B_T register

The "SS" portion of the first three commands is of no significance.

The other ten sets of index registers are used to modify the first two digits (SS) of the address portion of a command. In the discussion to follow, these index registers will carry the subscript (S), i.e., B_S, D_S, and L_S. The four commands that are used to modify the SS portion of a command are:

k	OP	ADDRESS	
k	73	SS00	Set B_S register
k	74	SS00	Set D_S register
k	75	SS00	Set L_S register
k	77	SSTT	Increment B_S register

The TT portion of the first three commands is of no significance.

A detailed description of the commands k7000TT, k7100TT, k7200TT and k76SSTT follows. The description of the commands k73SS00, k74SS00, k75SS00 and k77SSTT would be a repetion since these commands operate in exactly the same manner as the first four.

B REGISTERS

The index register B(k) is used to store a number which is added to the address of an order, before the order is obeyed, if the first digit in the command is (k). The order itself is left unmodified in the memory.

EXAMPLE: The index register $B_T(3)$ contains the number 18 and $B_S(3)$ contains 00. The command 0421720 replaces the contents of the AA register by the contents of address 1720. The order 3421720 replaces the contents of the AA register by the contents of address 1738. That is, the number 18 in index register $B_T(3)$ is added to 1720, the command address, before the order is obeyed. The order 3421720 remains in memory.

D REGISTERS

The index register D(k) is used to store a number which may be added to the contents of index register B(k).

L REGISTERS

The index register L(k) is used to store a number which limits the range through which the contents of the index register B(k) can be modified through successive additions of the contents of the index register D(k). A more detailed description follows.

Set B_T Register k7000TT

The command k7000TT places the number TT in the index register $B_T(k)$.

Example: After the command 37000u6 is obeyed, index register $B_T(3)$ contains the number u6.

Set D_T Register k7100TT

The command k7100TT places the number TT in the index register $D_T(k)$.

Example: After the command 4710028 is obeyed, the index register $D_T(4)$ contains the number 28.

Set L_T Register k7200TT

The command k7200TT places the number "TT" in the index register $L_T(k)$.

Example: After the command 6720003 is obeyed, the index register $L_T(6)$ contains the number 03.

Increment B_T Register k76SSTT

The command k76SSTT initiates the following sequence of events:

1. The number in $D_T(k)$ is added to the number in $B_T(k)$ and the results placed in $B_T(k)$.

2. If the new contents of $B_T(k)$ are less than or equal to the number in $L_T(k)$, control is transferred to the command in address SSTT.

3. If the new contents of $B_T(k)$ are greater than the number in $L_T(k)$, the next order in sequence is obeyed.

Example a: $B_T(1)$ contains 00; $D_T(1)$ contains 02; $L_T(1)$ contains 10; $B_S(1)$ contains 00; location 1320 contains the command 1761434. After this order is obeyed, the index register $B_T(1)$ contains 2. Since this is less than the number 10 in $L_T(1)$, control is transferred to the order in 1434.

Example b: $B_T(1)$ contains 10. Other conditions as in Example "a" after the order 1761434 in location 1320 is obeyed, the index register $B_T(1)$ contains 12, which is greater than the contents of $L_T(1)$; therefore, the next command comes from location 1321.

It is important to remember that anytime an index register (k) is named in a command this command will be modified by the contents of the B_T register and the B_S register.

The double index registers can be used to sum all of the numbers in Line 16 and Line 17. The following program is prepared as an example of the use of double index registers.

EXAMPLE

Bendix Computer

LOS ANGELES 45, CALIFORNIA

INTERCOM PROGRAM

Prepared by D. Hassell

Page ___ of ___

Date: 11-1-57

Line: 14

PROBLEM: Lines 16 and 17

NOTES	Location	K	OP CODE	Address		S/D	(A) or (AA)
Clears "AA"	1400		42	14	u0		0
Set B_T	1401	1	70	00	00		
Set D_T	1402	1	71	00	02		
Set L_T	1403	1	72	00	u6		
Set B_S	1404	1	73	00	00		
Set D_S	1405	1	74	01	00		
Set L_S	1406	1	75	01	00		
Before reaching 14.09 16.00+B_T After reaching 14.09 16+B_S.00+B_T	1407	1	43	16	00		
Increment B_T	1408	1	76	14	07		
Reset B_T = 0	1409	1	70	00	00		
Increment B_S	1410	1	77	14	07		
Type AA =	1411		33	21	00		Final
Return to Manual	1412		57	00	00	-	

BCD Form 108.12

Restriction on the use of the subroutine:

1. $0 \leq TT \leq u8$

2. $0 \leq SS \leq 31$

3. The routine must be stored in Line 09.

4. The routine is applicable to the G-15D only.

Any of the index registers may be modified using fixed point single precision arithmetic. The location of each of the registers is listed below.

LOCATION OF INDEX REGISTER

k	B_T	D_T	L_T	B_S	D_S	L_S
1	07.46	09.50	09.51	07.47	09.80	09.81
2	07.48	09.52	09.53	07.49	09.82	09.83
3	07.50	09.54	09.55	07.51	09.84	09.85
4	07.52	09.56	09.57	07.53	09.86	09.87
5	07.54	09.58	09.59	07.55	09.88	09.89
6	07.56	09.60	09.61	07.57	09.90	09.91
7	07.58	09.62	09.63	07.59	09.92	09.93
8	07.60	09.64	09.65	07.61	09.94	09.95
9	07.62	09.66	09.67	07.63	09.96	09.97
u	07.64	09.68	09.69	07.65	09.98	09.99

PART II

CARD INPUT-OUTPUT SUBROUTINE

With the addition of the card input-output subroutine for INTERCOM 1000 (G-15D) data can be read from IBM cards or punched on IBM cards. An IBM model 026 card punch and Bendix Computer Accessory CA-1 will be required to perform the above operation.

This subroutine must be stored in Line 10. This can be accomplished by executing the INTERCOM 1000 command 78 1000.

Read in Operation:

The input routine is designed to read four double precision floating point numbers, in the form of sign, 2 digits (characteristic) and 12 digits (mantissa). These four numbers should be placed in card columns 3-17, 21-35, 39-53 and 57-71.

The read program card for the 026 should be adjusted to fit the data cards. The requirement for the read program card are as follows:

1. Corresponding to a field on the data card, there should be a leading 2-punch (tab) two columns before the field, and a following 2-punch in the last column of the field.

2. There should be skips (single 11-punches, followed by a series of 12-punches) where necessary, through the second column preceding a leading 2-punch, and beginning the next column after the following 2-punch.

3. If more than two numbers are to be read, a 1-punch (reload) should precede the control punches of the next to last field. See NOTE page 8.

Read Data Card

The command 0081005 will cause the reading of one IBM card and will store the data. The first number is stored in Address 10.52,53; the second in 10.54,55; third in 10.56,57 and fourth in 10.58,59. First card should be registered in the read station prior to this command.

Punch Out Operation

The output routine is designed to punch on to IBM cards the information in the "AA" register. The number will be punched in the form of sign, 2 digits (characteristic) and 12 digits (Mantissa).

The punch program card should be prepared and inserted to the 026 prior to punching. Blank 15 column fields should be allowed in which the numbers are to be punched. The punch program card contains skips in all other columns.

In order to insure opportunity for the operator to change the program card in the IBM 026 between reading and punching operations, a halt command is available in the subroutine. To resume operation place the compute to idle then back to GO.

Punch on Card

The command 0081020 will cause a halt in a program to change a read--program to a punch program card.

The command 0081010 will cause the punching the contents of the "AA" register in a card field prescribed by the punch program card. The first card must be registered under the punch station.

NOTE:

If it is desired to read fewer than four numbers per card, "n", the number to be read per card should be stored as a hexadecimal constant at a scale factor of 2^{-8} in location 10.72 of the card input-output subroutine. For example 0300000 should be stored in 10.72 if three numbers are to be read from the card. Likewise, the "read" program card should be adjusted to correspond with the data card.

PART III

FLEXOWRITER INPUT SUBROUTINE

The Flex Input subroutine is written so that floating decimal numbers can be punched on a paper tape by an off line flexowriter model 34-5 and read in by INTERCOM 1000, converted and stored in Line 11 . . . ., 17 or 18.

This subroutine must be stored in Line 10. It used 50 word position, namely, 00 thru 49.

The entry to this command is 00810TT where TT is equal to the Line (11, 12, . . ., 17 or 18) in which input is to be stored.

Floating decimal numbers should be prepared in the following format:

```
 (tab) EEDDDDDDDDDDDD (sign) (tab) (tab) EEDDDDDDDDDDDD (sign) (CR)
/(tab) EEDDDDDDDDDDDD (sign) (tab) (tab) EEDDDDDDDDDDDD (sign) (CR)
   .        .           .      .     .         .          .     .
   .        .           .      .     .         .          .     .
   .        .           .      .     .         .          .     .
/(tab) EEDDDDDDDDDDDD (sign) (tab) (tab) EEDDDDDDDDDDDD (sign) (CR)
 s   (end of block)
```

NOTE: It is necessary to type sign of negative numbers only.

All blocks should contain less than 55 floating decimal numbers.

PART IV

TAPE FORMAT MODIFIER

The INTERCOM 1000 tape which makes reference to this supplement incorporates changes as outlined in Part V. These changes make the routine more powerful than the preliminary version. One basic change makes the routine approximately 50% faster when in "Automatic". Due to this change, tapes prepared by the preliminary INTERCOM 1000 cannot be read into the memory properly.

The use of the FORMAT MODIFIER will enable the read in of tapes prepared by the preliminary version. The method by which this can be accomplished follows:

1. Mount the tape format modifier on photo-reader.
2. Strike "p" key with enable on. (One block will be read)
3. Mount tape to be modified on photo-reader.
4. Move compute switch to BP.
5. The computer will read one block, modify, and punch out a new tape.
6. Move the compute switch to idle and back to BP. Another block will be read, modified and punched out, etc.
7. The modified tape can then be read into INTERCOM 1000.

NOTE:

A series of tests are made for floating point constants to prevent their conversion. There is a chance that these tests won't work for every case. Therefore, it is advised that a check be made of all constants on the modified tapes.

PART V

REVISIONS TO INTERCOM 1000

1. Variable Length Input (Floating Decimal Numbers)

Input of floating decimal numbers has been simplified by permitting the operator to type the significant digits of a floating decimal number only.

Example: To type in and store 51.123000000000 it suffices to type 51.123 tab s. The number will be converted, stored, reconverted, and typed out as 51.123000000000. To store zero type tab s and computer will respond by storing a zero in the designated location and typing back 50.000000000000. If an error is made before typing the tab s, the full fourteen digits must be typed.

2. Arithmetic Subroutines

a) The entries to all the arithmetic subroutines are the same as those described in Technical Memorandum 24A.

b) The following error halts and corrections have been made.

(i) Division by zero results in the following:

α. Rings bell

β. Halts on line 4. (If the compute switch is set to idle and then to GO, execution of commands will continue.)

(ii) To find source of error after Halt:

α. With compute on idle and enable on, type scf.

β. Put compute to GO and type -290000 to start selective print of the last command obeyed. This should point out the source of error.

c) Overflow. (When the result of a computation produces a result greater than 2^{128}). See 2b.

d) Underflow. (When the result of a computation produces a result less than 2^{-129}).

A zero will be placed in the "AA" register and computation will continue.

3. Additional OP Codes

a) Return to Auto (RTA) -0"60"0000

The computer will respond to the command -0600000 by starting computation on automatic at a point determined by the setting of the command counter.

This command is very useful when an operator desires to halt computation, transfer to manual and make several corrections. The operator doesn't have to know where he stopped computation to return to automatic at the correct place.

The address portion of the command is of no significance.

b) Ring Bell -0"33"0000

The operator responds to the command -0330000 by ringing the bell one time. The address portion of the command is of no significance.

c) (When INTERCOM 1000 is loaded on GO)

1)	K 7000TT	Set B_T
2)	K 7100TT	Set D_T
3)	K 7200TT	Set L_T
4)	K 73SS00	Set B_s
5)	K 74SS00	Set D_s
6)	K 75SS00	Set L_s
7)	K 76SSTT	Increment B_T
8)	K 77SSTT	Increment B_s

NOTE: For full description of above see Part 1 of this Supplement.

4. When on Selective print the contents of the "A" register will be typed every time a negative command is typed.

5. Computation speed has been increased by approximately 80% or 90% over its initial speed.

6. Type in Commands:

The computer will respond to the command -k45SSTT by typing SS.TT and setting type in. The operator types the command to be stored (BOPSSTT) followed by a tab s. The computer converts the command, types it in the format BTTOPSS, stores it in the designated address and types BTTOPSS followed by a carriage return.

Using the (-45SSTT) command, a fixed point sexadecimal number ($D_1D_2D_3D_4D_5D_6D_7$) must be typed in the format ($D_1D_4D_5D_6D_7D_2D_3$). The computer will convert the input and type it out in the converted form ($D_1D_2D_3D_4D_5D_6D_7$).

7. Loading instruction

a) Hit p with enable on and compute switch OFF

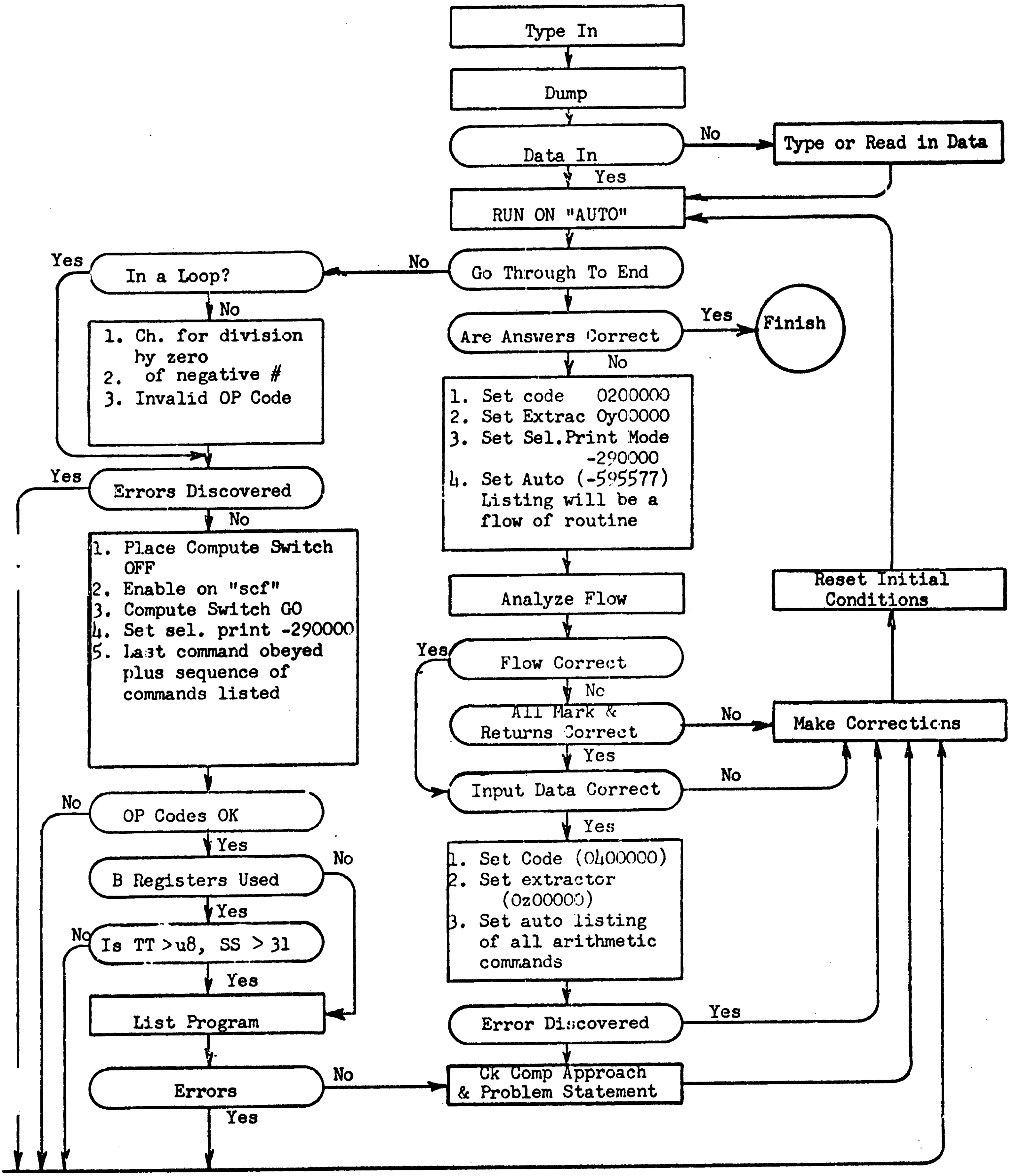
b) Put compute switch to GO (i) or BP (ii).

(i) Intercom 1000 plus the Automatic Index Register subroutine will be read into Lines 2, 3, 4, 5, 6, 7, 8, 9, and 0. "Op" codes listed under 3c above are valid.

(ii) When reading halts, with compute switch OFF and enable on hit f. The bell will ring signaling the completion of read in. "Op" codes listed under 3c above are invalid.

PART VI

DEBUGGING PROCEDURE FOR INTERCOM 1000



PART VII

INTERCOM 1000D - CHECK SUMS

The Intercom 1000D program consists of two tapes:

1) Basic Package 2) Subroutines

These tapes should contain the following check sums:

INTERCOM 1000D Basic Package

.40x7633
.0uu8vw6
-.u665054
.8919z28
-.3301vy8
-.3354305
-.y7w0121
.8w0y63x
.v7256uy
.5w4u59u
.68z20z1

INTERCOM 1000D Subroutines

-.15uw95z	Flex Input/Sq. Rt.
.87z9x83	Sin/Cos
-.6w70532	e^x, 2^x, 10^x
-.u46uyzu	$Log_e x$, $Log_{10} x$, $Log_2 x$
.ux7v086 ~~.15v8y56~~ ~~.22uv3y2~~	Card Input
-.6yy72z4	Tape Format Modifier

ACTION BY: All personnel concerned.

REFERENCES: Technical Applications Memorandum No. 24A.

REMARKS: Tapes will be sent to all G-15D installations approximately 15 November 1957

PREPARED BY: D. Hassell and T. Yamashita
[signature]

APPROVED BY: T. Yamashita
[signature]

TECHNICAL APPLICATIONS MEMORANDUM NO. 25

27 August 1957

TITLE: Flow Diagramming for the G-15 Computer

PURPOSE: To introduce the Bendix Template and to indicate some rules to follow when using the template

EQUIPMENT AFFECTED: All

EFFECTIVE DATE: 27 August 1957

INSTRUCTION:

Flow diagramming has always been of enormous help to engineers, programmers and others who believe in planning their problems in advance. Since with this memorandum the Bendix Flow Charting Template is being introduced, it is intended to present a set of rules, which would be of assistance when using the template. Thus flow diagrams made by one individual would be understood by another, and also the amount of time required to code the problem would be minimised since all the logical errors should be easily detected from the flow diagram.

Essentially the flow control of any routine is marked by the following:

1. Operations to be performed in a certain order

2. Branching points or points of conditional transfer of control

3. Merging points

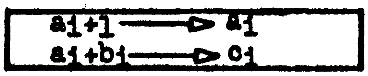
4. Remote connections, which may either be fixed or variable

The symbols on the Bendix Template are summarised in the Appendix A of this memorandum.

There are three types of operation boxes; all of them enclose a sequence of operations performed in a certain specified order and the only difference between them is their position in the routine, i.e.:

This box is always placed at the beginning of the routine.

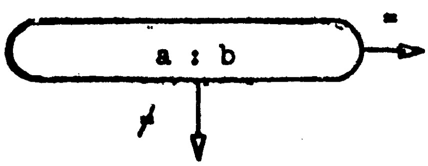
This box is always placed at the end of the routine.



This box is used in all other situations.

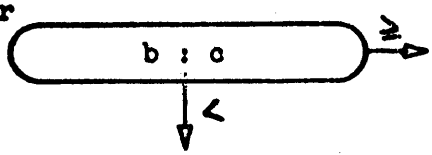
Thus in any of the above boxes all operations performed, e.g. arithmetic calculations, should be enclosed in their proper order.

The next type of operations would be the conditional transfer of control, where the flow of operations branches out upon certain conditions being present or absent.

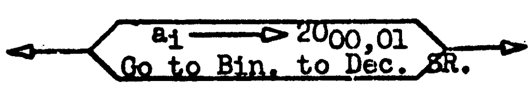


This box is being used for conditional transfer of control tests.

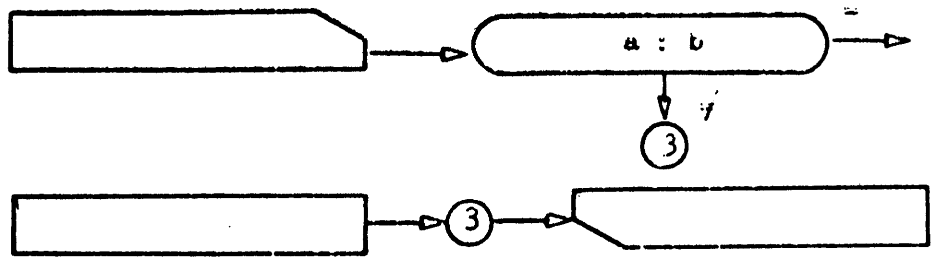
or



It often happens that many auxiliary operations are being performed by a subroutine, and in order to follow the main trend it has been found convenient to indicate such operations by using the following symbol:



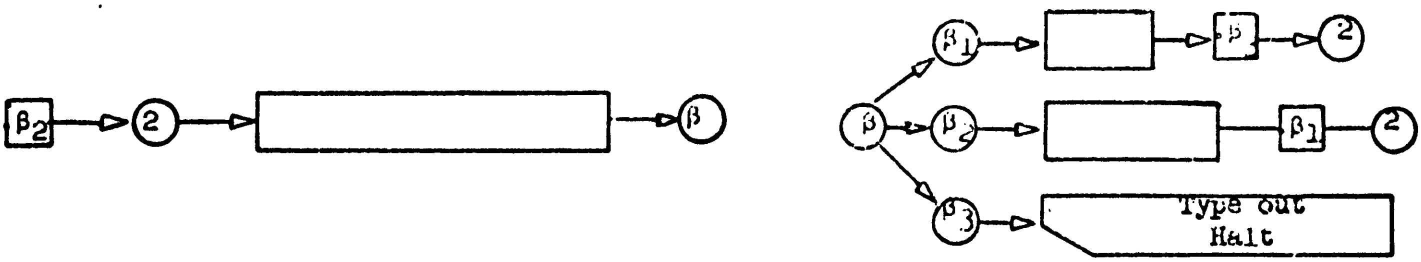
The next symbol will indicate remote connections which will be either fixed or variable. It may happen that as a result of a conditional test we would like to jump into another part of the program. Thus instead of interweaving a line in a spider-web manner through the whole program a remote connector should be used instead:



(3) - This being a remote connector, which will provide for a jump from one part of the program to another.

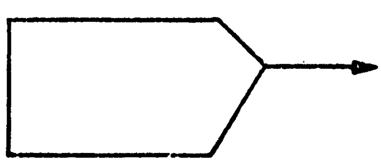
The variable connectors are by convention designated with the Greek letters, and since in many cases it is desired to select one of several possibilities, the required connection must be "set" beforehand, e.g. [β_1]

This would indicate that β_1 connection was selected and set.



From the above diagram it can be seen that the first pass would be through β_2, where after some operation β_1 connection is set. The next pass would be through β_1, where this time after some operation β_3 connection was set. The final pass would be through β_3 to the end of the program.

The two remaining symbols on the Bendix Template refer to DA-1 programs:



This box represents an integrator or a decision element.

This box represents a constant multiplier.

It is felt that with the prudent use of the above symbols flow charting of all Bendix Computer programs will become easy and understandable to all who prepare computer programs.

ACTION BY: All personnel concerned

REFERENCES: None

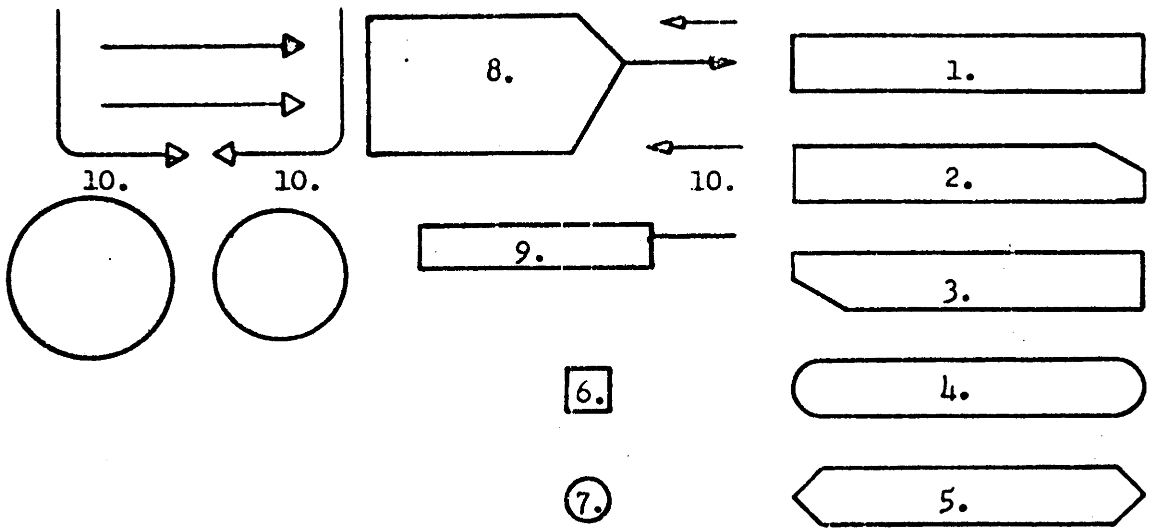
REMARKS: As agreed at the First Bendix Computer Users' Conference

PREPARED BY: Z. Jelinski

APPROVED BY: T. Yamashita

ZJ/TY/bs

FLOW DIAGRAMMING SYMBOLS FOR BENDIX COMPUTER PROGRAMS



1. Operation box enclosing operations to be performed in specified order
2. Operation box always placed at the beginning of a routine
3. Operation box always placed at the end of a routine
4. Box enclosing logical operations resulting in branching
5. Box in the main program indicating the use of subroutine
6. Box for setting variable connector
7. Circle for fixed or variable connection
8. DA-1 Integrator or Decision Box
9. Constant Multiplier
10. Direction of flow control

TECHNICAL APPLICATIONS MEMORANDUM NO. 28

29 August 1957

TITLE: DAPPER-1A

PURPOSE: Description of Revisions to DAPPER-1

EQUIPMENT AFFECTED: G-15D, DA-1

EFFECTIVE DATE: 29 August 1957

INSTRUCTION:

Minor revisions have been incorporated in DAPPER-1 and provision added for a DA-1 memory dump to punched tape and subsequent reloading.

I. Corrections:

A. Two commands in the DAPPER-1 loader have been modified on tape, in accordance with the original coding, to permit the correct set-up for 27 or 54 integrator operation.

B. The overflow indicator is turned off in response to op code 7, freeing op code 9 for use as described below. The overflow detection block has been rewritten to conserve commands and provide space in the same line for most of the memory dump program.

C. In case of overflow in an integrator, the code digits "zz" are typed out in place of "v", to avoid confusion with an op code. Other typeouts after overflow are as before.

D. Additional delay has been introduced in the DAPPER-1A loader to ensure the typeout of the number of integrators set up.

II. Punched Tape Memory Dump:

A. Description

After the memory dump block has been read from the DAPPER-1A tape, a leader is punched out and then another block of DAPPER-1A tape is read into Line 19. This block contains the skeleton of a loading routine, into which are stored the contents of the z lines (21 and 22), the typeout list, check sums of all other lines to be punched, and information designating which integrator (if any) has been coded for halt control and which line contains initial conditions (Line 05 for 108 integrators, Line 10 for less). The check sum of Line 19 is then adjusted to zero and the line is punched.

If the DA-1 problem to be dumped has been set up for 108 - integrator operation, lines 05 through 18 are punched; if fewer integrators were specified, lines 07 through 18 are punched.

The memory dump preserves the contents of Y, R, K, and K_r registers, as well as initial conditions and address lines. A problem may thus be dumped for later reloading and continuation either from the point at which calculations halted or after a return to initial conditions.

The contents of the dX and dY counters and registers are not recorded on punched tape. Consequently, on continuation of an interrupted DA-1 problem after memory reload, inputs addressed to integrators 00, 01, 02, 03 may be incorrect during the first cycle only.

Since the contents of the Y registers are disturbed in memory during punching, the dump routine exits through the DAPPER-1A program to insert initial conditions (op code y).

B. Operation

After a problem has been loaded from the typewriter, whether or not initial conditions have been inserted (op code y) or calculations performed (op code x), a type in of 9 (tab) s initiates the memory dump. A bell rings as the last block is being punched.

The process takes 10 to 12 minutes, so it is not recommended for short one-shot jobs.

III. Memory Reload:

A. Description

The DAPPER-1A program must have been stored before loading DA-1 memory from punched tape.

The first block on a memory dump tape is a self-checking loader for the blocks which follow. If its check sum differs from zero, the bell rings and computation halts.

After each subsequent block is read, its check sum is compared with that obtained before the tape was punched. In case of error, the tape is reversed, the bell rings, and the block reread.

At the conclusion of loading memory with a DA-1 problem, control is returned to the DAPPER-1A program for typed input.

B. Operation

1. Load DAPPER-1A in the usual way.

2. Whether DA-1 calculations have been performed or not, or whether DA-1 is on or not, turn the COMPUTE switch OFF, then remove the DAPPER-1A tape without rewinding.

3. With a problem tape in the reader, use "p" to read its loader, then COMPUTE to GO.

4. Replace the DAPPER-1A tape magazine.

5. Check the command line switches at the rear of DA-1, to be sure they provide enough address lines for the new problem.

6. To insert initial conditions, type "y (tab) s".

7. To resume DA-1 computations from the point where memory was dumped, omit step 6 and type "x (tab) s".

ACTION BY: All personnel concerned

REFERENCES: DA-1 Programming Manual
Technical Applications Memorandum No. 21

REMARKS: None

PREPARED BY: S. H. Lewis

APPROVED BY: T. Yamashita

TECHNICAL APPLICATIONS MEMORANDUM NO. 29

27 September 1957

TITLE: Programming and Operational Differences between the G-15A and the G-15D.

PURPOSE: To summarize the differences between the two G-15 models for the benefit of the user of one machine who receives programs written for the other, or for the G-15A user who contemplates the acquisition of a G-15D. Familiarity of the reader with one G-15 model or the other is assumed.

EQUIPMENT AFFECTED: G-15A and G-15D.

EFFECTIVE DATE: 27 September 1957

INSTRUCTION:

Notation

The notation used below for commands conforms with the Programmers Reference Book and Bendix Coding Manuals of current issue; i.e., the form is that accepted by the PPR conversion routines for both the G-15A and G-15D.

A decimal command format now obsolete was formerly used with the G-15A before the introduction of PPR. The programmer specified for each command whether its execution was to be immediate or deferred, and wrote for T the decimal equivalent of the binary T in the converted command. The effect was the same as though every command in the PPR format were written with a prefix u or w. The N number was designated T_n, and three digits were required for all location, T, and T_n numbers. The addition of 4 to a characteristic signified a break point rather than double precision, which was specified by a negative sign at the end of the command.

1. Input and Output Processes and Control

1.1 Accessory Equipment

The only accessories available for use with the G-15A are magnetic tape units. Therefore, the G-15A has no commands corresponding to those in the G-15D for controlling or testing the status of a differential analyzer, punched card equipment, fast tape punch, or special input and output registers. Commands with D=31 and S=02, 03, 14, 18, or 19 have no effect in the G-15A. Other G-15D commands which refer to accessories are present in the G-15A, but with different functions. The commands with S=11, D=31 (punch cards) and S=29, D $<$ 31 (20·IR) are discussed below under topic 1 (1.3), "Punching and Typeout", and topic 5, "Extract Operations", respectively. In commands with S=17 or 28 and D=31, the C code is immaterial in the G-15A; however, these commands should be written with C=0 to correspond with the G-15D.

1.2 Functions of four-word lines

Input data entered into either computer are stored in a four-word line until a "reload" signal causes the contents of this line to proceed, via a second four-word line, to line 19. In the G-15A, the first four-word line, designated V, is not addressable by program as a source or destination. On the reload, the contents of line V are copied into line 23 before entry into line 19, but they are not retained in the intermediate line. Therefore, input data become accessible to a G-15A program only after reaching line 19.

In the G-15D, however, input data enter line 23 first, where up to four of the last words entered are available to commands with S=23. Moreover, line 23, serving as a destination, can be cleared by a program before input so that leading zeros need not always be typed into the G-15D. The buffer line, intermediate between line 23 and line 19 for G-15D input performs its function automatically and is not accessible to either programmed or manual operations.

During a process of output from the G-15A, the contents of line V are modified. The details of this activity need not concern the programmer, except that he can not expect any specific part of line V to be clear before a subsequent input operation. The corresponding function in output from the G-15D does not involve line 23, but is performed in the intermediate buffer line mentioned above.

The contents of G-15A lines 22 and 23 are replaced during output to magnetic tapes; only line 23 is involved in the G-15D. No disturbance of addressable four-word lines occurs in other outputs.

Some confusion may arise from the retention in certain publications of an obsolete notation for the four-word lines in the G-15A, where the Roman Numerals I, II, III and IV have been used in reference to lines 20, 21, 22 and 23, respectively. The perpetuation of this anachronism is to be condemned and discouraged.

1.3 Punching and Typeout

The commands with S=08, 09, or 10 and D=31, to initiate separate typeout or tape punching, are the same in both the G-15A and the G-15D. Simultaneous output to both the typewriter and tape punch is possible from either model of the computer, but the control of the process differs in the two machines.

In the G-15A, the contents of line 19 are typed and punched concurrently in response to a command with S=11 and D=31. The C code is immaterial and one word time is sufficient for the execution of the command. In the G-15D, however, this command initiates output from line 19 to an auxiliary card punch.

The contents of either AR or line 19 of the G-15D will be punched on tape while being typed out if the PUNCH switch on the typewriter base is <u>on</u>. That is, for simultaneous output to the typewriter and the tape punch, the switch must have been set manually by the time the typeout command is executed. This switch is normally <u>off</u>, since no punching is desired with most typeouts. In order to coordinate the setting of the PUNCH switch with automatic computation, the program must include a command to test the position of the switch and suspend computation until the switch is turned to the proper position.

The position of the G-15D PUNCH switch is tested by a command of the form L_2 N 1 17 31. (If L_1 is used in place of L_2, the bell will also sound, regardless of the outcome of the test!) The succeeding command is read from location N if the switch is <u>off</u>, from N+1 if the switch is <u>on</u>.

Case 1. L_0) L_2 L_0 1 17 31

L_1) L_3 N 0 09 31

The test performed by the command in L_0 will be repeated if the PUNCH switch is off. After the switch is turned on, the command in L_1 is executed and the contents of line 19 will be simultaneously typed and punched. If the switch is turned off during the output process, the punch will stop before the end of the record, but the typeout will continue to completion.

Case 2.	L_0)	L_2	L_{-1}	1	17	31
	L_{-1})	L_1	N	0	09	31

If the command in L_0 is executed first, the test will be repeated while the PUNCH switch is on. Typeout of (19) will occur when the switch is turned off.

The PUNCH switch on the G-15A serves only to feed tape.

On the G-15D only, a typeout may be prevented or interrupted by turning the ENABLE switch on. This permits the adjustment of paper, stopping output by typing "s" without jamming keys, etc. If output is not stopped by "s", the commanded typeout will resume when the ENABLE switch is turned off. An output to the punch, in response to a command with S=10 and D=31, will not begin while the ENABLE switch is on; although the switch does not interrupt a punch-out in progress.

One other typeout operation shows a slight difference between the G-15A and the G-15D. The "t" key, in manual control of either machine, replaces the high order eight bits of AR by the N number of the last command executed. If N=00, however, the eight bits in the G-15A are represented in hex by 14; in the G-15D they are 94, to avoid ambiguity with the case where N=20 (decimal).

1.4 Setting "Ready"

When input is being read from the G-15A typewriter, the effect of the "s" key depends on the state of the ENABLE switch. If the switch is on, the "Ready" state is set without reload of the contents of line V into line 19; if the switch is off, not only is the "Ready" state set, but the reload operation also occurs.

The "s" signal after type-in to the G-15D does not cause a reload from line 23 to line 19, regardless of the position of the ENABLE switch. Normally, fewer than four words of information are typed in before "s" is struck, and the input is available directly from line 23.

"Ready" may be set by a command in a G-15D program, but not in the G-15A. The G-15D command with S=00 and D=31 will stop any input or output operation; and in addition, it may disturb the contents of line 19. A precise determination of the effect on line 19 depends on detailed knowledge of both the input-output circuits and a number of conditions at the time the command is executed.

Two examples of G-15D programming illustrate the usefullness of the "Set Ready" Command. First, it becomes possible to eliminate the use of the "s" key after a type-in to line 23. A binary 1 is stored in this line in every fourth bit position before the type-in command is executed. Each digit typed in precesses the line by four bits, so no change in the contents of the line will be apparent ahead of the entered digits. If the type-in ends with a tab or carriage return, however, the line is precessed only one bit; upon detection of the changed configuration, the "Set Ready" command is executed.

A second example of the application of the "Set Ready" command in the G-15D occurs in tape punching, where a format of sign codes is used to punch leaders between blocks and a condensed format of 29 digits and end code is used for punching significant information. With line 19 clear, the leader is punched by giving a command of the form 01 N 0 10 31 <u>every</u> drum cycle, tallying until approximately 144 drum cycles have occurred, then executing a "Set Ready" command. After "Ready" has been set, the punch format is stored in line 02, the line to be punched is copied into line 19, and the punch command executed. (Ref. Technical Applications Memorandum No. 20).

1.5 <u>Punched Tape Reversal</u>

A G-15A command with S=07, D=31, causes the punched tape reader to reverse until a stop code is detected. Thus, to prepare to read tape from a position n blocks earlier than the last previous stop, it is necessary to program n+1 reversals and one tape read (S=15, D=31), each followed by a test for "Ready".

A G-15D command with S=06, D=31, causes the punched tape reader to reverse past two stop codes and then move forward once. Each execution of this command moves the tape backwards to a position where it is prepared for reading one block earlier. If a command with S=07, D=31, is executed, the tape will move back and forth once, stopping in its original position.

On the G-15D typewriter only, the "b" key ("Ready" state, ENABLE switch <u>on</u>) initiates the same punched tape reversal as the S06, D31 command.

2. Command Lines

2.1 Lines Available

The only G-15A memory lines, except AR, from which commands can be read for execution are 00, 01, 19, and V. In the G-15D, the eight lines 00, 01, 02, 03, 04, 05, 19, and 23 are all available.

Differences between the two G-15 models in both manual and programmed selection of a command line are discussed below. In both machines, the use of AR as a command source is the same. After the execution of a command in AR, control is returned to whichever command line was previously active, unless the command in AR itself transfers control to another line.

2.2 Manual Selection

A G-15A switch, not present on the G-15D, provides the only means for selecting either line 19 or line V as a source of commands. The left hand setting of this three-position switch causes commands to be read from line 19; the right hand setting makes V the command line. The central position of the COMMAND switch specifies either line 00 or line 01, the choice being determined by the execution of a programmed command and indicated by a neon lamp on the indicator panel. If either line 19 or line V is selected by the switch setting, the condition of the indicator is immaterial. The execution of a command to set the indicator off (line 00) or on (line 01) activates the designated line as a command source only after the switch has been returned to the central position.

The manual "mark place" operation ("m" key, ENABLE on), stores an indication of the command line selector (0 or 1) in the G-15A, but not in the G-15D. On "return" ("r" key, ENABLE on), the G-15A selector is restored to the condition that existed at the time of marking, but no change occurs in the G-15D command line. If the last G-15A command executed before a "mark place" was Multiply, Divide, Shift, or Normalize, part of the contents of AR will not be restored on "return". This loss does not occur in the G-15D. Otherwise, the "mark" and "return" operations are the same in both models of the G-15.

When the G-15A COMMAND switch is set to select line V, single cycle operation results from the "e" key (ENABLE on); when the COMMAND switch is set for line 19 or in the central position, the "i" key (ENABLE on) causes one command to be executed. On the G-15D, the "e" key has no effect; the "i" key (ENABLE on) is used for single cycle operation in any command line.

In the G-15D, any of the eight command lines may be selected manually by the use of typewriter control keys, with the selection indicated in binary form by three lamps on the panel. With the computer in the "Ready" state, the "c" key (ENABLE on) clears the indicator, selecting line 00 for commands. Another line may be chosen by striking the corresponding digit key (1, . . ., 5 for lines 01, . . ., 05; 6 for line 19, or 7 for line 23) with the ENABLE switch on. Only when the G-15D is in the "Ready" state does the ENABLE switch cause a digit key to influence command line selection. Moreover, a digit key may turn indicator lamps on; but only the "c" key turns them off. For example if line 05 is to be selected and the "4" key is struck by mistake, the correction can be made by following with either "1" or "5". However, if the operator's aim were off to the right and "6" had been hit instead of "5", it would be necessary to use "c" before the error could be corrected.

Line 23 is automatically selected as the G-15D command source in two cases, both of which involve the reading of punched tape. At the completion of the DC power turn-on cycle (topic 6, below), or after punched tape reading has been initiated with the "p" key, not only is line 23 selected and indicated as the command source, but also the first command will be read from 23.00 without the need for use of the "f" key. There is no change in command line or sequence when punched tape is read in response to a programmed command (S 15, D 31).

2.3 Selection by Program

In the G-15A, a command of the form L_2 N 0 20 31 causes the next command to be read from line 00, word N, when the COMMAND switch is in the central position. Line 01 is selected by a similar command with S=21. Although these commands are normally written with C=0, the characteristic is actually immaterial; and the word position of the next command is unconditionally N.

G-15D commands with S=20 or 21 and D=31 also select a command line; but here the line is specified by the C code, and the commands differ in their manner of determining the word position of the next command.

A G-15D command of the form w T N C 21 31 transfers control unconditionally to word N of line C. (C=0, . . ., 5 for lines 00, . . ., 05; C=6 for line 19; C=7 for line 23.) Also, during the one word time in which the command is executed, the number T is stored in the command register, marking a word position to which control may later be returned.

A command, in the G-15D, of the form L_2 L_1 C 20 31 transfers control to the line specified by C, but to the word position last marked by a command with S=21, D=31. In general, if the "return" command is written L_2 N C 20 31 and the last marked T occurs earlier, after L_1, than N, the next command is read from word position T. If, however, $L_2 \leq N \leq T$, the next command comes from word N. A break point in a G-15D command with S=20, D=31, is ignored; and in single cycle operation both this and the following command are executed in response to one operation of the "i" key.

3. Division Command and Overflow Detection

The G-15A command for division (S=25, D=31) is normally written with T an even number, and the C code (for conversion by PPR-A1) must be 5 (i.e., double precision, ch = 1). There is no automatic detection of overflow; but if the first quotient bit developed (bit 1 of MQ odd if T = 58) is not zero, the quotient is greater than or equal to one. After a double precision division (T = u6), the first quotient bit is lost and no examination for overflow is possible.

Automatic setting of the overflow indicator after improper division is provided in the G-15D. In the divide command, either C = 1 or C = 5 is permitted. Whether single or double precision overflow is to be detected depends on whether the T number in the command is odd or even.

If T is odd, the quotient bit developed in position 29 of MQ even during the next-to-last cycle is sent to the overflow indicator and is not recorded in bit position 1 of MQ odd, since the execution of the command ceases at the end of an even word time. (The location of the divide command being odd, both the first and last times of execution are even.) Thus, for normal single precision division, T = 57 and the overflow indicator is set unless the quotient is a fraction.

If T is even in a G-15D command for division, the bit sent to the overflow indicator is that developed in position 29 of MQ odd during the next-to-last cycle. For example, if T = v6, a double precision quotient will be formed and overflow indicated if the result is not fractional. If T = 58, as in normal G-15A single precision division, an overflow indication may occur in the G-15D due to a bit which is not part of the developed quotient, or no indication may occur when an overflow exists.

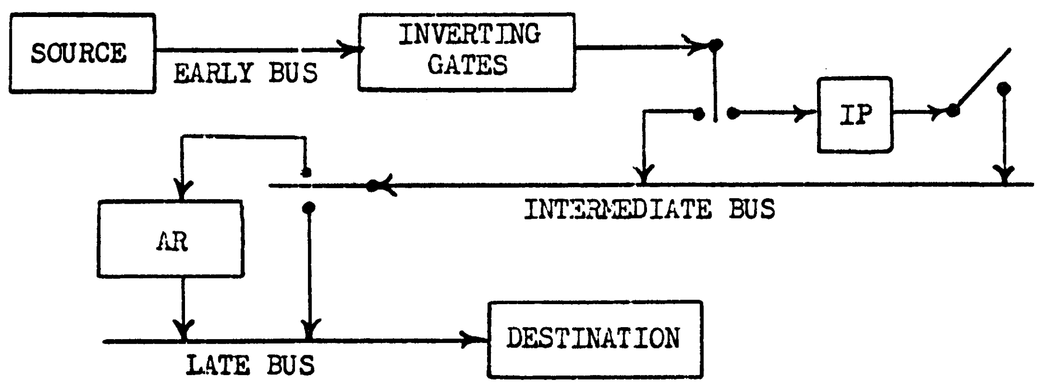
4. Information Copies between Two-word Registers

When information is sent from storage into MQ, ID, or PN, or when information is stored from a two-word register, the effects of various C codes and the handling of signs are the same in both the G-15A and the G-15D. A difference between the two machines is effective only when the C code is even in a command having a two-word register as source and the same or another two-word register as destination.

In the G-15A, the rules governing the function of IP in copies between two-word registers are derived by simultaneous application of the rules governing the case where one of the registers is a source and the rules for the case where a two-word register is a destination.

In the G-15D, the rules are more simple; when information is copied between two-word registers, IP is not involved; if C is even, absolute values are copied. The only exception occurs when C = 0 or 4 and S = D = 26 (PN), as described below.

The following diagram and summary of rules describe the behavior of IP in both the G-15A and G-15D, with differences noted where they exist.



<u>IP Flip-Flop</u>

a. If C = 0 or C = 4; or if C = 2 or C = 6 and S < 28; then (IP) is <u>changed</u>, regardless of source in the G-15A, or if S ≠ 24, 25, or 26 in the G-15D:

  i. From 0 to 1, or from 1 to 0, by a negative sign on the early bus if D = 24 (MQ) or 26 (PN)

  ii. To agree with a sign on the early bus if D = 25 (ID)

b. (IP) is made 0 by command 0 23 31 (clear).

c. (IP) is <u>unchanged</u> otherwise [eg., if C is odd, or if C = 2 or 6 and S ≥ 28 (abs, value), if D = 30 (PN+), if S = 24, 25, 26 (G-15D only), etc.]

d. If S = 24 (MQ), or 25 (ID), or 26 (PN), regardless of destination in the G-15A but only if D ≠ 24, 25, 26, or 27 in the G-15D, then (IP) enters the sign bit on the intermediate bus either if C = 0 or 4, or if C = 2 or 6 and D < 28.

e. In the G-15D only, a command having C = 0 or 4 and S = D = 26 (PN) copies (IP) into the TE bit of PN and complements (PN) if negative. (IP) is unchanged.

NOTE: A sign is placed on the early bus from Bit 1 of a single word source or from TE (Bit 1, even word) of a double word source, including MQ, ID, or PN. If S (or D) is 24 (MQ), 25 (ID), or 26 (PN), and if C = 0 or 4, or if C = 2 or 6 and D (or S) < 28, then the sign bit does not reach the intermediate bus from the early bus.

5. Extract Operations

The extract operations with S = 27, 30 or 31 and D < 31 are the same in both G-15A and G-15D.

In the G-15A, a command with S = 29 and D < 31 forms the source number from (20 + 21); that is, the logical inclusive "or". Since the operation specified by S = 27, D < 31, has proven far more useful, the former command has been assigned to a different function in the G-15D.

A G-15D command with S = 29, D < 31 operates on the contents of a special input register (IR) which may be attached as optional auxiliary equipment. The word formed as source is (20·IR). If no such special register is used, S = 29 may be used to provide an unconditional source of zero.

Two extract operations are available in the G-15D for which no counterparts exist in the G-15A. A command with S = 18 and D = 31 transmits (20·ID) to a special output register which may be attached to the G-15D as optional auxiliary equipment. A command of the form w T N 3 (or 7) 23 31 operates on the contents of ID and PN with reference to an extractor stored in line 02. This command copies (02·PN) into ID, leaving ($\overline{02}$·PN) in PN. Although introduced to permit binary to decimal conversion during DA-1 operation, the command facilitates this conversion whether the differential analyzer is in use or not.

6. Turn-on Cycle

6.1 Number Track

The number tracks differ in word u7, where the G-15A has -1414000 and the G-15D has -1414794. The corresponding check sums are 147y000 for the G-15A and 147x86w for the G-15D.

In the G-15A, the number track, once entered from punched tape and stored during the DC turn-on cycle, is not accessible by any programmed command. In the G-15D, however, the command L_1 N 1 31 31 causes the stored number track to be copied into line 18, where it may be inspected for verification. The command actually causes the contents of the number track to be superimposed on the contents of line 18 which should, therefore, have been previously cleared.

6.2 Loading Routine

When DC power is turned on, the G-15A reads one block of tape and stores the contents in the number track. The next block, usually a "bootstrap" or loading routine, must be read by manipulation of the "p" key, and control established at word 00 of of line 19 by use of the "f" key and setting of the COMMAND switch.

The turn-on of DC power in the G-15D not only causes the number track to be read from tape and stored, but also initiates the reading of a second tape block and establishes control at 23.00 automatically.

All loading routines written for use on a G-15D should include, early in the program, a command of the form 00 N 1 19 31 (Stop DA-1), to insure that no portion of the G-15D memory assigned for use in DA-1 operation will be inactive.

Loading routines for the G-15A which are to be executed from line 0, should contain a break point or halt command, to allow the command source switch to be placed in the center position.

ACTION BY: All G-15 users

REFERENCES: G-15 Programmers Reference Book, G-15A Coding Manual, G-15D Coding Manual, Technical Applications Memorandums No.'s 8 and 14

REMARKS: None

PREPARED BY: S. H. Lewis S. H. Lewis

APPROVED BY: T. Yamashita T. Yamashita

HL/TY/bs

TECHNICAL APPLICATIONS MEMORANDUM NO. 30

11 June 1958

TITLE: Changes in G-15D Logic - Shift Command

PURPOSE: To provide for more flexibility in programming

EQUIPMENT AFFECTED: G-15D from #100

EFFECTIVE DATE: 21 March 1958

INSTRUCTION: Effective G-15 Serial Number #100, command characteristic other than "0" will block the tally operation in the accumulator when shifting or normalizing. This information has not been made generally known to all persons affected by the change. This change is of such a nature that it can easily be added to all machines, from 0 to 99 by the Customer Service Group.

ACTION BY: All personnel concerned

REFERENCES:

REMARKS: None

PREPARED BY: T. Yamashita
APPROVED BY: T. Yamashita and F. Adair

TECHNICAL APPLICATIONS MEMORANDUM NO. 32

4 November 1957

TITLE: Fixed Point, Double Precision Subroutines

PURPOSE: To provide advance information on double precision, fixed point subroutines

EQUIPMENT AFFECTED: G-15A and G-15D

INSTRUCTION:

The following specifications for the fixed point, double precision subroutines are being issued as appendices to this Technical Applications Memorandum:

1.	Arctangent	Appendix A
2.	Square Root	Appendix B
3.	Sine - Cosine	Appendix C

ACTION BY: All personnel concerned

REFERENCES: None

REMARKS: The arctangent and square root subroutines are on the same block of tape. The sine-cosine subroutine is on another. These subroutines were extracted from a traverse closure program.

PREPARED BY: A. J. da Costa *A J da Costa*

APPROVED BY: T. Yamashita

APPENDIX A

Arctangent R

Range of argument R:	$0 \leq R < 1$ (See Note 1)
Output:	Arctan R in circles and in radians
Execution:	From Line 1
Data Input:	(a) Return command in AR;
	(b) Argument (double precision) in $ID_{0,1}$ (See Note 2)
Entry:	At word time 72 (See Note 2)
Output Locations:	(a) Arctan R in circles in $PN_{0,1}$;
	(b) Arctan R in radians in $20_{0,1}$
Exit:	The return command is read at word time 73.
Short Lines Used:	Entire lines 20, 21.
Accuracy:	Correct to the nearest millionth of a second. (error $< 5 \cdot 10^{-7}$ second)
Execution Time:	11.05 drum cycles (321 milliseconds).
Error Stops:	None

NOTE 1: If the argument $R > 1$, enter $1/R$ as argument and subtract the result from 1/4 circle.

If $R = 1$, arctan $R = 1/8$ circle (that is $\pi/4$ radians).

NOTE 2: Entry may be made with the argument in MQ, but the entry location is then at word time 68. Care should then be taken that the two-word registers hold the correct sign.

REFERENCE: "Table of Arctan x" National Bureau of Standards, Applied Mathematics Series 26.

Method:

The method is based on the formula $\arctan R - \arctan L = \arctan \frac{R - L}{1+RL}$, where L is a number conveniently close, and involves a "table look-up" for arctan L and the determination by Maclaurin's series of the reduced argument $u = \frac{R - L}{1+RL}$

It consists of five steps:

(1) The arctangents of the eighths of unity: 0, 1/8, 1/4, ... 1, taken from a twelve-figure table (see reference) are stored in the computer.

(2) That eighth (L) which is closest to the argument R is determined. We have then $R - L = 1/16$.

(3) Arctan $\frac{R - L}{1+RL}$ is calculated by Maclaurin's series: $\arctan u = u - \frac{u^3}{3} + \frac{u^5}{5} - \frac{u^7}{7}$, where $u = \frac{R-L}{1+RL}$.

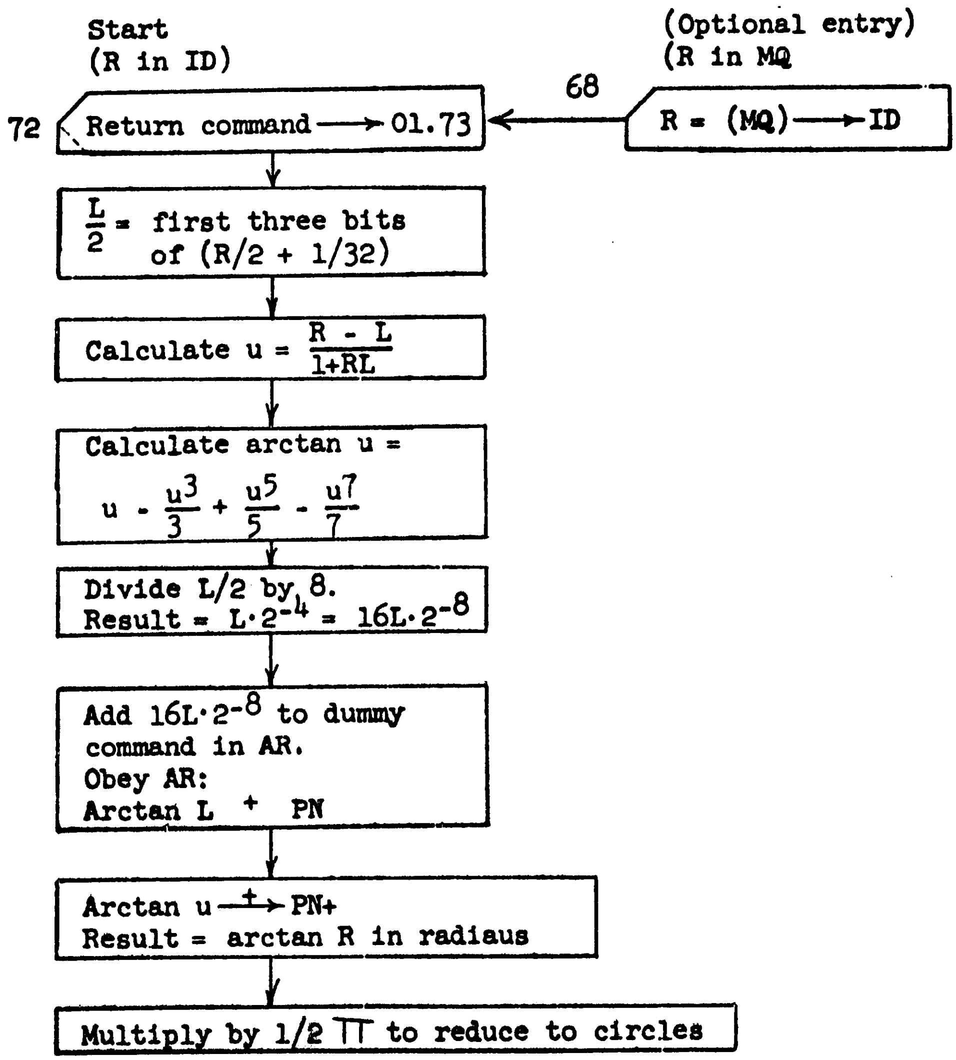
Four terms are sufficient since $u \leq 1/16$.

(4) The appropriate arctan L is selected by a "table look-up", clued by the value of L.

(5) Finally, arctan R = arctan L + arctan u.

FLOW CHART

Arctan R



Bendix Computer
Los Angeles 45, California

Page 1 of 1
Date: 7-1-57
Line 01

G-15D
Prepared by A. J. da Costa
PROGRAM PROBLEM: SQUARE ROOT - DOUBLE PRECISION, FIXED POINT

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35
36	37	38	39
40	41	42	43
44	45	46	47
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63
64	65	66	67
68	69	70	71
72	73	74	75
76	77	78	79
80	81	82	83
84	85	86	87
88	89	90	91
92	93	94	95
96	97	98	99
u0	u1	u2	u3
u4	u5	u6	

L	P	T or L_k	N	C	S	D	BP	NOTES
u6		00	23	5	26	20		N = (PN) $\xrightarrow{+}$ 20.00,01
23		26	31	0	28	01		Ret. Cmd. = (AR)→01.26
31		34	34	0	23	31		Clear
34		40	51	4	01	25		$1 - 2^{-58}$ (01.40,41) → ID
40		}	hex					zzzzzzz } $1 - 2^{-58}$
41								-zzzzzzz
51		02	56	0	26	31		Shift 1 pl. {(ID) $\frac{r_i}{2}$}
56		60	63	4	20	26		N = (20.00,01)→PN
63		v2	69	5	25	31		Divide (112 w.t.)
								{$(MQ)=\frac{1}{4}\left(\frac{N}{r_i/2}\right) = \frac{N}{2r_i}$}
69		70	86	4	24	26		$\frac{N}{2r_i}$ = (MQ)→PN
86		88	93	7	25	30		$\frac{r_i}{2}$ = (ID) $\xrightarrow{-}$ PN+
93		94	u5	1	26	28		$r_{i+1} - r_i$ = (PN_0) $\xrightarrow{+}$ AR
								{For sign test in Cmd. 18}
u5	u	02	18	5	25	30		$\frac{r_i}{2}$ = (ID) $\xrightarrow{+}$ PN + (4w.t.)
								{(PN) = r_{i+1}}
18		19	26	-	22	31		(AR) < 0 ?
26								≥ 0 Ret. Cmd.
27		28	51	4	26	25		< 0 r_{i+1} = (PN)→ID

FORM BCD 108-5.0

APPENDIX B

Square Root

Argument:	N, $0 \leqslant N < 1$
Output:	$\sqrt{N}$ (Positive Root)
Execution:	From Line 1
Data Input:	(a) Return command in AR;
	(b) Argument N (double precision) in $ID_{0,1}$
Entry:	At word time 106 (u6)
Output Locations:	(a) $\sqrt{N}$ in $PN_{0,1}$
	(b) $\frac{1}{2}\sqrt{N}$ in $ID_{0,1}$
Exit:	The return command is read at word time 26.
Short Line Usage:	Only words 0 and 1 of Line 20.
Accuracy:	Error $< 2^{-56}$ (That is, correct to the 16th decimal place)
Error Stops:	None
Execution Time:	Variable (Approximately 1 drum cycle for each leading binary zero.)

NOTE: Execution time may be saved by a preliminary test for $N = 0$.

Method:

The argument A, in circles, is first reduced to u, where u is the positive difference between A and the nearest quarter of a circle, expressed in radians.

Thus $0 \leqslant u \leqslant \pi/4$

Then Maclaurin's expansion is used for both functions:

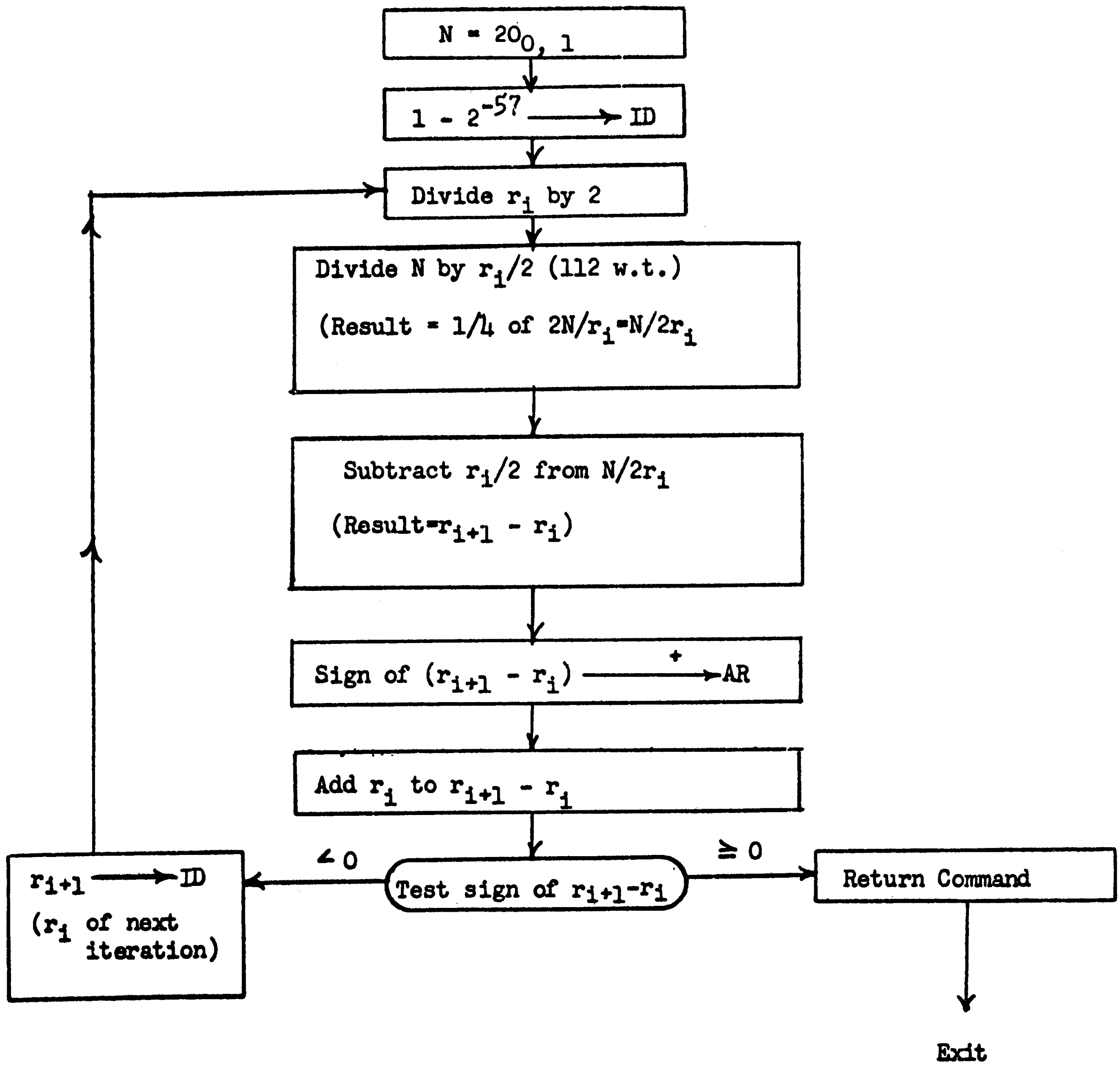
$$\sin u = u - \frac{u^3}{3!} + \frac{u^5}{5!} - \dots + \frac{u^{13}}{13!}$$

$$\cos u = 1 - \frac{u^2}{2!} + \frac{u^4}{4!} - \dots + \frac{u^{12}}{12!}$$

Seven terms in each series are sufficient to ensure accuracy to 12 decimal digits.

As the outputs are $\frac{1}{2}$ sin A and $\frac{1}{2}$ cos A, the coefficients are divided by 2 in the program.

FLOW CHART


N = 20$_{0,1}$
$1 - 2^{-57}$ → ID
Divide r_i by 2
Divide N by $r_i/2$ (112 w.t.)
(Result = 1/4 of $2N/r_i = N/2r_i$
Subtract $r_i/2$ from $N/2r_i$
(Result = $r_{i+1} - r_i$)
Sign of $(r_{i+1} - r_i)$ + → AR
Add r_i to $r_{i+1} - r_i$
Test sign of $r_{i+1} - r_i$
< 0
≥ 0
r_{i+1} → ID
(r_i of next iteration)
Return Command
Exit

Bendix Computer

Los Angeles 45, California

Page 1 of 4
Date: 6-26-57
Line 01

G-15D

Prepared by A. J. da Costa

PROGRAM PROBLEM: ARCTAN R, DOUBLE PRECISION, FIXED POINT

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35
36	37	38	39
40	41	42	43
44	45	46	47
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63
64	65	66	67
68	69	70	71
72	73	74	75
76	77	78	79
80	81	82	83
84	85	86	87
88	89	90	91
92	93	94	95
96	97	98	99
u0	u1	u2	u3
u4	u5	u6	

L	P	T or L_k	N	C	S	D	BP	NOTES
00		(hex)	000	0	000			$\tan^{-1} 0 = 0$
01		(hex)	000	0	000			
02		(hex)	558	y	529			$\tan^{-1}(\frac{1}{8})$ = .124, 354, 994, 547
03		(hex)	-1zx	5	vu9			(decimal)
04		(hex)	4v2	4	zyz			$\tan^{-1}(\frac{1}{4})$ = .244, 978, 663, 127
05		(hex)	3yv	6	yvz			
06		(hex)	z27	y	855			$\tan^{-1}(\frac{3}{8})$ = .358, 770, 670, 271
07		(hex)	5vx	8	650			
08		(hex)	voy	4	u71			$\tan^{-1}(\frac{1}{2})$ = .463, 647, 609, 001
09		(hex)	76v	1	9w1			
10		(hex)	xz0	y	193			$\tan^{-1}(\frac{5}{8})$ = .558, 599, 315, 344
11		(hex)	-8z0	0	5x5			
12		(hex)	269	4	xz9			$\tan^{-1}(\frac{3}{4})$ = .643, 501, 108, 793
13		(hex)	-u4v	w	7x1			
14		(hex)	785	3	69x			$\tan^{-1}(\frac{7}{8})$ = .718, 829, 999, 622
15		(hex)	-v80	5	3y2			
16		(hex)	441	x	525			$\tan^{-1}(1)$ = .785.398, 163, 397
17		(hex)	w90	z	xuu			
72		73	74	0	28	01		Return Command ⟶ 01.73
74		76	79	4	25	20		R = (ID) ⟶ $20_{0,1}$
79		02	82	0	26	31		Shift (Divide by 2)
82		83	84	0	01	26		1/32 ⟶ PN_1
83	<	(hex)	080	0	000		>	= 1/32
84		86	89	4	25	30		R/2 ⟶ PN+

FORM BCD 108-5.0

Bendix Computer
Los Angeles 45, California

Page 2 of 4
Date: 6-26-57
Line 01

G-15D
PROGRAM PROBLEM: ARCTAN R, DOUBLE PRECISION FIXED POINT

Prepared by A. J. da Costa

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35
36	37	38	39
40	41	42	43
44	45	46	47
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63
64	65	66	67
68	69	70	71
72	73	74	75
76	77	78	79
80	81	82	83
84	85	86	87
88	89	90	91
92	93	94	95
96	97	98	99
u0	u1	u2	u3
u4	u5	u6	

L	P	T or L_k	N	C	S	D	BP	NOTES
89		90	92	4	26	21		$(R/2) + (1/32) \longrightarrow 21_{2,3}$
92		94	96	4	01	20		Extractors $\longrightarrow 20_{2,3}$
94	<	(hex)	000	0	000		>	Extractors
95	<	(hex)	z00	0	000		>	
96		98	u0	4	31	20		$L/2 \longrightarrow 20_{2,3}$
u0		u2	u4	4	25	26		$R/2 \longrightarrow$ PN
u4		u6	20	7	20	30		Subtract L/2 from R/2
20		22	24	5	26	21		$1/2(R-L) \xrightarrow{+} 21_{2,3}$
24		26	30	4	20	25		$L/2 = (20_{2,3}) \longrightarrow$ ID
30		32	35	4	20	24		$R = (20_{0,1}) \longrightarrow$ MQ
35		v4	42	0	24	31		Multiply L/2 by R
42		44	46	4	01	30		Add 1/2 to RL/2
44	<	(hex)	000	0	000		>	= 1/2 (double precision)
45	<	(hex)	800	0	000		>	
46		48	52	4	26	25		$1/2\ (1 + RL) \longrightarrow$ ID
52		54	57	4	21	26		$1/2(R - L) \longrightarrow$ PN
57		v6	66	5	25	31		Divide (T = 116)
66		68	70	4	24	20		$u = \frac{R - L}{1+RL} \longrightarrow 20_{0,1}$
70		72	75	4	20	25		$u \longrightarrow$ ID
75		76	81	4	20	24		$u \longrightarrow$ MQ
81		v4	90	0	24	31		Multiply
90		92	97	4	26	21		$u^2 \longrightarrow 21_{0,1}$
97		98	u1	4	01	25		$1/7 \longrightarrow$ ID
98	<	(hex)	924	9	249	>		1/7=(dec) .142, 857, 142, 857, 14
99	<	(hex)	249	2	492	>		
u1		u4	21	4	21	24		$u^2 \longrightarrow$ MQ

Bendix Computer
Los Angeles 45, California

Page 3 of 4
Date: 6-26-57
Line 01

G-15D
Prepared by A. J. da Costa

PROGRAM PROBLEM: ARCTAN R, DOUBLE PRECISION FIXED POINT

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35
36	37	38	39
40	41	42	43
44	45	46	47
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63
64	65	66	67
68	69	70	71
72	73	74	75
76	77	78	79
80	81	82	83
84	85	86	87
88	89	90	91
92	93	94	95
96	97	98	99
u0	u1	u2	u3
u4	u5	u6	

L	P	T or L_k	N	C	S	D	BP	NOTES
21		v4	28	[illegible]	24	31		Multiply 1/7 by u^2
28		30	32	4	26	21		$u^2/7 \longrightarrow 21_{2,3}$
32		36	38	4	01	26		$1/5 \longrightarrow$ PN
36	<	(hex)	666	6	667		>	} 1/5
37	<	(hex)	333	3	333		>	}
38		72	47	7	21	30		$u^2/7 \xrightarrow{-}$ PN+
47		48	50	1	26	24		$1/5 - u^2/7 \longrightarrow$ MQ
50		52	55	4	21	25		$u^2 \longrightarrow$ ID
55		v4	64	[illegible]	[illegible]	31		Multiply
64		60	41	[illegible]	26	21		$u^2/5 - u^4/7 \longrightarrow 21_{2,3}$
71		76	78	4	01	26		$1/3 \longrightarrow$ PN
76	<	(hex)	uuu	u	8[illegible]v		>	} 1/3
77	<	(hex)	555	5	555		>	}
78		82	85	7	21	30		$u^2/5 - u^4/7 \xrightarrow{-}$ PN+
85		86	91	4	26	24		$1/3 - u^2/5 + u^4/7 \longrightarrow$ MQ
91		92	u3	4	21	25		$u^2 \longrightarrow$ ID
u3		v4	19	0	[illegible]	31		Multiply
19		20	22	4	26	25		$u^2/3 - u^4/5 + u^6/7 \longrightarrow$ ID
22		24	25	7	25	30		$u^2/3 - u^4/5 + u^6/7 \xrightarrow{-}$ PN+
25		26	29	4	26	24		$1 - u^2/3 + u^4/5 - u^6/7 \longrightarrow$ MQ
29		32	39	4	20	25		$u \longrightarrow$ ID
39		v4	48	0	24	31		Multiply
48		52	54	4	26	20		Arctan $u \longrightarrow 20_{0,1}$
54		55	59	0	20	25		$L/2 \longrightarrow ID_1$
59		56	67	0	26	31		Shift 3 places (Div. by 8)
67		69	80	0	25	28		$L \cdot 2^{-4} = (ID) \longrightarrow$ AR

Bendix Computer Los Angeles 45, California

Page 4 of 4

Date: 6-26-57

Line 01

G-15D

Prepared by A. J. da Costa

PROGRAM PROBLEM: ARCTAN R, DOUBLE PRECISION FIXED POINT

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35
36	37	38	39
40	41	42	43
44	45	46	47
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63
64	65	66	67
68	69	70	71
72	73	74	75
76	77	78	79
80	81	82	83
84	85	86	87
88	89	90	91
92	93	94	95
96	97	98	99
u0	u1	u2	u3
u4	u5	u6	

L	P	T or L_k	N	C	S	D	BP	NOTES
80		88	u2	0	01	29		Dummy Sel. Comm. ⟶ AR+
88	<	00	43	5	01	26	>	Dummy Sel. Comm. read at T=u6
u2		u5	u6	0	31	31		Obey AR (Sel. Comm)
(AR)		(16L	43	5	01	26		Arctan L = (01.16L) $\xrightarrow{+}$ PN
43		44	49	5	20	30		Arctan u = ($20_{0,1}$) $\xrightarrow{+}$ PN+
49		50	53	4	26	25		Arctan R (Radians) ⟶ ID
53		56	58	4	25	20		Arctan R (Radians) ⟶ $20_{0,1}$
58		60	65	4	01	24		1/2π ⟶ MQ
60	<	(hex	727	2	4uv		>	1/2π = .159,154,943,091,90
61	<	(hex	-28v	y	60x		>	
65		v4	73	0	24	31		Multiply (to reduce to circles)
								[Arctan R(circles) in PN]
73								Return Command
68		70	72	4	24	25		R = (MQ) ⟶ ID (Optional MQ entry)

APPENDIX C

Sine and Cosine

Argument:	Angle A as a fraction of a circle. $0 \leq A < 1$
Output:	$\frac{1}{2}$ sin A and $\frac{1}{2}$ cos A.
Execution:	From Line 01
Input Data:	(a) Return command in AR;
	(b) Argument A (Double Precision) in $PN_{0,1}$
Entry:	At word time 30
Scaling of Input:	The argument A must be expressed as a part of a circle. That is, before entry to the subroutine, if A is in degrees it must be divided by 360, or if A is in radians it must be divided by 2π.
Output Data:	$\frac{1}{2}$ sin A in $22_{0,1}$
	$\frac{1}{2}$ cos A in $22_{2,3}$
Exit:	The return command is read at word time 31.
Short Lines Used:	Entire Lines 21, 22 and words 01, 03 of Line 20.
Accuracy:	Correct to 12 decimal places (Error $< 5.10^{-13}$)
Execution Time:	34 drum cycles (986 milliseconds)

NOTE: Both $\frac{1}{2}$ sin A and $\frac{1}{2}$ cos A are obtained with one entry of the argument. This is of great advantage in many problems.

Method:

The argument A, in circles, is first reduced to u, where u is the positive difference between A and the nearest quarter of a circle, expressed in radians.

Thus $0 \leqslant u \leqslant \pi/4$

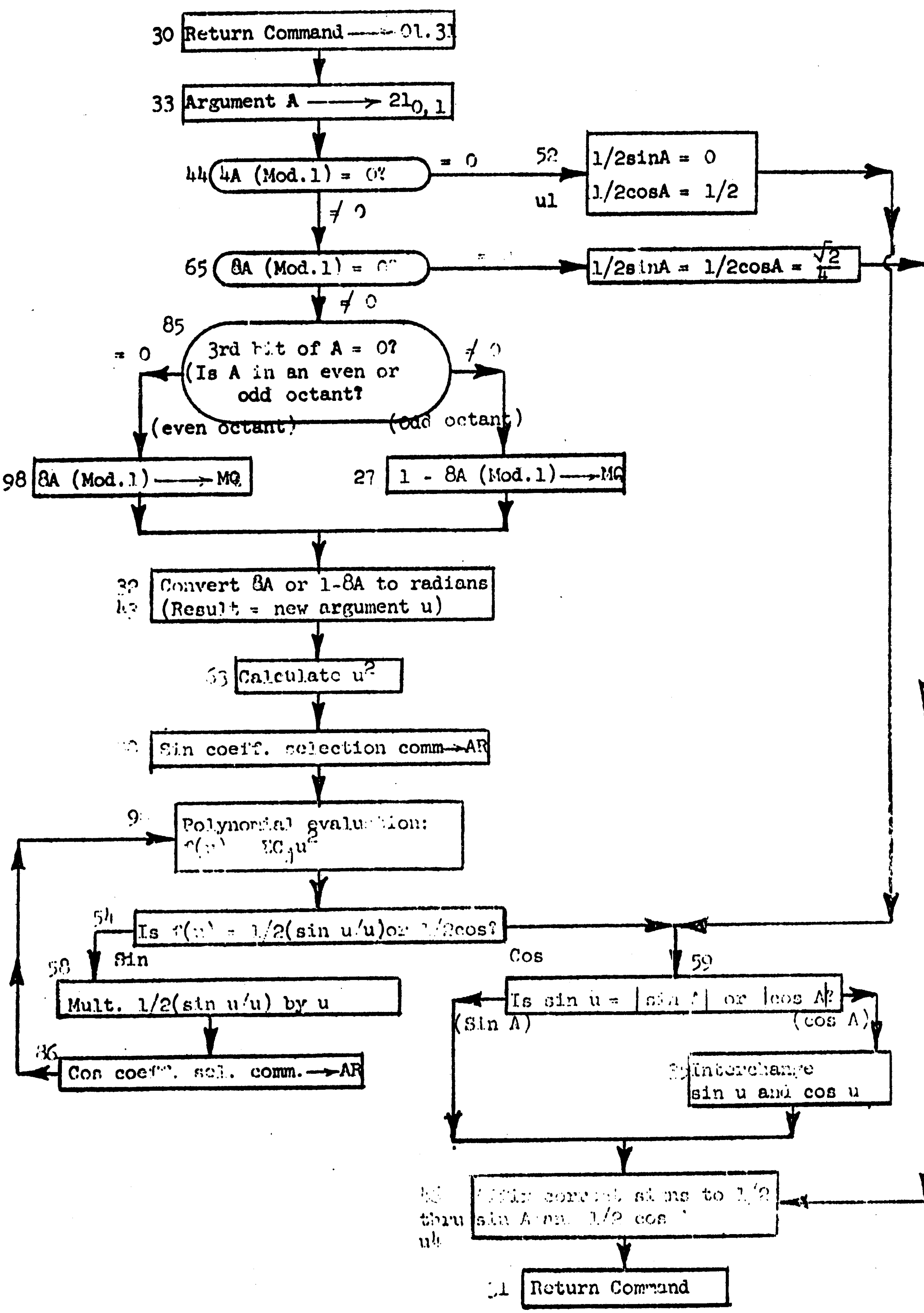
Then Maclaurin's expansion is used for both functions:

$$\sin u = u - \frac{u^3}{3!} + \frac{u^5}{5!} - \dots + \frac{u^{13}}{13!}$$

$$\cos u = 1 - \frac{u^2}{2!} + \frac{u^4}{4!} - \dots + \frac{u^{12}}{12!}$$

Seven terms in each series are sufficient to ensure accuracy to 12 decimal digits.

As the outputs are $\frac{1}{2}\sin A$ and $\frac{1}{2}\cos A$, the coefficients are divided by 2 in the program.


30 Return Command → 01.31
33 Argument A → $21_{0,1}$
44 4A (Mod.1) = 0?
= 0
≠ 0
52
ul
1/2sinA = 0
1/2cosA = 1/2
65 8A (Mod.1) = 0?
1/2sinA = 1/2cosA = $\frac{\sqrt{2}}{4}$
85
3rd bit of A = 0?
(Is A in an even or odd octant?
= 0
(even octant)
(odd octant)
98 8A (Mod.1) → MQ
27 1 - 8A (Mod.1) → MQ
Convert 8A or 1-8A to radians
(Result = new argument u)
Calculate u^2
Sin coeff. selection comm → AR
Polynomial evaluation:
54
Is f(u) = 1/2(sin u/u) or 1/2cos?
Sin
Cos
58
Mult. 1/2(sin u/u) by u
86
Cos coeff. sel. comm. → AR
59
(Sin A)
(cos A)
Interchange
sin u and cos u
thru
Return Command

Bendix Computer Los Angeles 45, California

G-15 D

Page 1 of 5

Date: 6-14-57

Line 01

Prepared by A. J. da Costa

PROGRAM PROBLEM: SINE - COSINE DOUBLE PRECISION, FIXED POINT

				L	P	T or L_k	N	C	S	D	BP	NOTES
0	1	2	3									
4	5	6	7	30		31	33	0	28	01		Return Cmd. = (AR)→01.31
8	9	10	11	33		36	39	4	26	21		Argument A→$21_{0,1}$
12	13	14	15	39	u	44	44	4	26	30		(PN)→PN + (4 word times) (= 4 times A)
16	17	18	19	44		46	52	4	26	27		4 A (modulo 1) = 0 ?
20	21	22	23	52		56	u1	4	26	22		= 0 ½ sin u = 0 →$22_{0,1}$
24	25	26	27	u1		u6	70	4	01	22		½ cos u = ½ →$22_{2,3}$
28	29	30	31	53		54	65	4	26	30		≠ 0 Add |4A| to itself
32	33	34	35	65		66	68	4	26	27		8A (modulo 1) 0 ?
36	37	38	39	68		80	93	4	01	25		= 0 $\sqrt{2}/4$ = (01.80,81)→ID
40	41	42	43	81								(hex) - 5u82700 } $\sqrt{2}/4$
44	45	46	47	80								(hex) 3z9xw47 }
48	49	50	51	93	u	98	70	4	25	22		$\sqrt{2}/4$→Line 22 (4 w.t.)
52	53	54	55									(sin u = cos u = $\sqrt{2}/2$)
56	57	58	59	69		70	74	4	26	21		≠ 0 8A (mod. 1)→$21_{2,3}$
60	61	62	63	74		77	85	0	01	20		Extractor = (01.77)→20_1
64	65	66	67	77								(hex) 2000000 = extractor (=1/8)
68	69	70	71	85		89	98	0	31	27		3rd bit of A = 0 ? {Even or add octant?
72	73	74	75	98		uz	32	4	21	24		= 0) Even octant 1 8A (mod. 1)→MQ
76	77	78	79	99		uo	u5	4	25	25		≠ 0) odd octant 1 (ID)→ID to clear PN
80	81	82	83	u5		u6	27	7	21	30		8A→PN + (to complement 8A)
84	85	86	87	27		28	32	4	26	24		1 - 8A→MQ
88	89	90	91	32		36	43	4	01	25		$\frac{\pi}{4}$→ID
92	93	94	95	37								(hex) w90zxuu } π/4
96	97	98	99	36								(hex) 442x279 }
u0	u1	u2	u3	43		v4	57	0	24	31		Multiply to reduce to radians
u4	u5	u6										New argument u = 2πA or π/4−2πA (modulo π/4)

Bendix Computer

Los Angeles 45, California

G-15D

Prepared by A. J. da Costa

Page 2 of 5

Date: 6-14-57

Line 01

PROGRAM PROBLEM: SINE - COSINE DOUBLE PRECISION, FIXED POINT

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35
36	37	38	39
40	41	42	43
44	45	46	47
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63
64	65	66	67
68	69	70	71
72	73	74	75
76	77	78	79
80	81	82	83
84	85	86	87
88	89	90	91
92	93	94	95
96	97	98	99
u0	u1	u2	u3
u4	u5	u6	

L	P	T or L_k	N	C	S	D	BP	NOTES
57		60	63	4	26	22		u → $22_{0,1}$
63		64	67	4	22	25		u → ID
67		68	71	4	25	24		u → MQ
71		v4	79	0	24	31		Multiply: (PN) = u^2
79		82	88	4	26	22		u^2 → $22_{2,3}$
88		89	90	0	01	28		Sin. Coeff. Sel. Comd. → AR
< 89	u	26	29	4	01	26	>	C_{13} = (01.24,25) → PN
90	u	94	94	0	23	31		Clear
94		98	u3	4	25	21		Clear $21_{2,3}$
u3		u5	u5	0	31	31		Obey AR ←
< (AR)	u	(2j)	29	4	01	26	>	Cj → PN
29		30	38	7	21	30		Subtract f_{j+1} from Cj
								(Cj - $f_{j+1} = g_j$)
38		40	45	3	01	29		Subtract 4 from timing number of Sel. Cmd.
40								(hex) 0400000
45		47	48	0	22	31		(AR) > 0 ? All coefficients completed?
49		50	61	4	26	24		Neg. g_j → MQ
61		62	75	4	22	25		u^2 → ID
75		v4	91	0	24	31		Multiply (Double Precision)
91		94	u3	4	26	21		$f_j = u^2 g_j$ → $21_{2,3}$
								Pos. Exit from loop
48		50	54	3	01	29		Constant = (01.50) → AR+
50								(hex) zy1x03u = constant to discriminate between sine and cosine evaluation.
54		56	58	0	28	27		(AR = 0 ? (Sine or cos ?)
								(= 0) Sin: Evaluation completed

FORM BCD 108-5.0

Bendix Computer Los Angeles 45, California

Page 3 of 5
Date: 6-14-57
Line 01

Prepared by A. J. da Costa

G-15D
PROGRAM PROBLEM: SINE - COSINE DOUBLE PRECISION, FIXED POINT

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35
36	37	38	39
40	41	42	43
44	45	46	47
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63
64	65	66	67
68	69	70	71
72	73	74	75
76	77	78	79
80	81	82	83
84	85	86	87
88	89	90	91
92	93	94	95
96	97	98	99
u0	u1	u2	u3
u4	u5	u6	

L	P	T or L_k	N	C	S	D	BP	NOTES
58		60	64	4	26	25		sin u/2u ⟶ ID
64		68	73	4	22	24		u = $(22_{0,1})$ ⟶ MQ
73		v4	82	0	24	31		Multiply: (PN) = ½ sin u
82		84	86	4	26	22		½ sin u ⟶ $22_{0,1}$
86		87	90	0	01	28		Cos.Coefficients Sel. Cmd. ⟶ AR
<87	u	24	29	4	01	26	>	C_{12} = (01.22,23) ⟶ PN
								(Back to Cmd. 90 for polynomial evaluation)
								(≠0) (Cos. evaluation completed)
59		62	70	4	26	22		½ cos u ⟶ $22_{2,3}$
70		73	76	2	21	28		Abs. val. of most sig. digits ⟶ AR
76		77	97	0	01	29		Add 1/8 = (01.77)
97		99	u2	0	28	21		|A| + 1/8 ⟶ 21_3
u2		03	28	0	01	20		Extractor 4000000 ⟶ 20_3
28		31	34	0	31	27		2nd bit of A = 0 ?
34		36	46	0	00	00		(=0) sin A = sin u (Skip to Cmd. 46)
35	u	46	46	2	22	22		(≠0) sin A = cos u cos A = sin u
								T.V.A. 10 w.t. to interchange sin u and cos u
46		48	51	4	22	25		|½ sin A| ⟶ ID
51		53	55	4	25	27		½ sin A = 0 ?
55		57	72	0	00	00		(=0) Skip to 72
56		57	60	2	21	28		(≠0) |A| ⟶ AR
60		61	62	0	28	29		(AR) ⟶ AR + (to double |A|)
62		64	66	0	28	24		(AR) ⟶ MQ (sign ⟶ ID)
66		68	72	4	25	22		½ sin A = (ID) ⟶ $22_{0,1}$

Bendix Computer

Los Angeles 45, California

Page 4 of 5
Date: 6-14-57
Line 01

G-15 D

Prepared by A. J. da Costa

PROGRAM PROBLEM: SINE - COSINE DOUBLE PRECISION, FIXED POINT

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35
36	37	38	39
40	41	42	43
44	45	46	47
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63
64	65	66	67
68	69	70	71
72	73	74	75
76	77	78	79
80	81	82	83
84	85	86	87
88	89	90	91
92	93	94	95
96	97	98	99
u0	u1	u2	u3
u4	u5	u6	

L	P	T or L_k	N	C	S	D	BP	NOTES
72		74	78	4	22	25		$\lvert \frac{1}{2} \cos A \rvert = (22_{2,3}) \rightarrow$ ID
78		82	83	4	25	27		$\frac{1}{2} \cos A = 0$?
83		86	31	0	00	00		(=0) Skip to Cmd. 31
84		85	92	2	21	28		$(\neq 0)$ $\lvert A \rvert \rightarrow$ AR
92		95	96	9	20	29		Add $\frac{1}{1}$ circle
96		97	u0	0	28	29		Add $\lvert A \rvert + \frac{1}{4}$ to itself
u0		u2	u4	0	28	24		(AR) $\rightarrow$ MQ (Sign $\rightarrow$ ID)
u4		u6	26	4	25	22		$\frac{1}{2} \cos A \rightarrow 22_{2,3}$
26		28	41	0	29	31		Reset overflow
41		44	31	0	00	00		Skip to Cmd. 31
42		44	31	0	00	00		Skip to Cmd. 31
31								Return Command
				COEFFICIENTS OF POLYNOMIALS				
								(hex)
u6								0000000 } $C_0 = \frac{1}{2}$
u7								8000000 }
00								0000000 } $C_1 = \frac{1}{2}$
01								8000000 }
02								0000000 } $C_2 = 1/2 \cdot 2!$
03								4000000 }
04								uuuuuuv } $C_3 = 1/2 \cdot 3!$
05								1555555 }
06								uuuuuuv } $C_4 = 1/2 \cdot 4!$
07								0555555 }

FORM BCD 108-5.0

Bendix Computer

Los Angeles 45, California

Page 5 of 5

Date: 6-14-57

Line 01

G-15D

Prepared by A. J. da Costa

PROGRAM PROBLEM: SINE - COSINE DOUBLE PRECISION, FIXED POINT

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35
36	37	38	39
40	41	42	43
44	45	46	47
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63
64	65	66	67
68	69	70	71
72	73	74	75
76	77	78	79
80	81	82	83
84	85	86	87
88	89	90	91
92	93	94	95
96	97	98	99
u0	u1	u2	u3
u4	u5	u6	

L	P	T or L_k	N	C	S	D	BP	NOTES	
08								2222222	$C_5 = 1/2 \cdot 5!$
09								0111111	
10								05v05v0	$C_6 = 1/2 \cdot 6!$
11								-002x82x	
12								00x00x0	$C_7 = 1/2 \cdot 7!$
13								-0006806	
14								u01u01u	$C_8 = 1/2 \cdot 8!$
15								-0000x00	
16								vw74uuy	$C_9 = 1/2 \cdot 9!$
17								-0000171	
18								z93yxxu	$C_{10} = 1/2 \cdot 10!$
19								-00000z4	
20								v9915u0	$C_{11} = 1/2 \cdot 11!$
21								0000003	
22								8z76w78	$C_{12} = 1/2 \cdot 12!$
23								0000000	
24								0v09230	$C_{13} = 1/2 \cdot 13!$
25								0000000	

FORM BCD

TECHNICAL APPLICATIONS MEMORANDUM NO. 33

18 November 1957

TITLE: Programming Output to Punched Tape

PURPOSE: To provide information concerning formats and timing in programs for punching leader and data on tape.

EQUIPMENT AFFECTED: G-15A, G-15D, Flexowriter

EFFECTIVE DATE: 18 November 1957

INSTRUCTION:

I. General Discussion

In punching blocks of data on tape and leader between blocks, the choice of format and the length of leader required may depend upon whether the tape is prepared solely for reentry into the G-15, for transcription on a printing device such as a Flexowriter, or for both. This memorandum discusses programming considerations and describes two sequences of commands for the automatic punching of a leader of any desired length, followed by the punching of a record in any of several formats.

On tapes prepared for entry into the G-15, experience has shown that each block of data should be preceded by a leader at least 7 inches long. Occasionally a shorter leader may suffice, or a slightly longer one be required, depending on mechanical adjustments in an individual photo reader. For quick visual recognition the leader would ideally be blank except for sprocket holes; however, negative signs, periods, or even zero digits do not interfere with proper data reading and the verification of check sums.

Leader may be punched automatically under the control of only the most significant format character stored in line 02 if a punch command of the form 01.N.0.10.31 is executed either in consecutive drum cycles or in alternate cycles, or if a punch command of the form W.00.N.0.10.31 is executed in consecutive drum cycles. One punch stroke occurs for every two drum cycles, so a desired length of leader may be obtained by programming a count of 20 drum cycles or 10 punch strokes per inch of leader.

Tapes prepared only for transcription on a Flexowriter require little or no leader between data blocks.

The data format on tapes for Flexowriter printing, or those punched with simultaneous G-15 typeout, must include such control characters as periods, tabs, carriage returns, etc. However, if nothing but the reentry of data into the G-15 is intended, then reading time, tape stock and, especially, punching time are conserved when the format is more condensed. The most economical data format consists of 29 digits and an end (reload) code. This format is not suitable for punching leader unless line 19 is clear and a G-15D "Set Ready" command is executed before output data is copied into line 19.

II. Single Format Punching

The simplest program punches both leader and data with a single format. This must begin with a sign code, which may be followed either by 29 digits and an end code or by other characters if required to control simultaneous or subsequent typing.

The following typical sequence of commands may be executed on either G-15A or G-15D with output data stored in line 19 and any format beginning with a sign code in the proper words of line 02. The ratio of the constants C_1 and C_2 determines the length of leader.

Location	T	N	C	S	D	Notes
N_0	T_0	N_1	0	S	28	$C_1 > 0 \rightarrow$ AR
N_1	T_1	N_2	3	S	29	$C_2 > 0 \Rightarrow$ AR+
N_2	T_2	N_3	0	22	31	Sign (AR) $\rightarrow$ Test
N_3	01	N_1	0	10	31	Punch
$N_3 + 1$	Anything except input/output					

If $00 < N_1 < T_1 < N_2 < T_2 - 1 < N_3 < u7$, then the punch command occurs in consecutive drum cycles and the length of leader is approximately $.05C_1/C_2$ inches. If the timing is such that the punch command is executed in alternate drum cycles, the leader length is approximately $C_1/10C_2$ inches. Minus signs are punched in the leader if (19.u7) is negative.

III. Punching with change of Format

Although a sign code should be used for punching leader, it may be desirable to punch data in a format beginning with a digit or other character. The format codes stored in word 03 of line 02 may be changed between leader and data punching, with no interruption of punch cycles, if any one of the following timing conditions is satisfied.

(a) The format (02.03) may be changed between the last two in a sequence of punch commands of the form 01.N.0.10.31, executed in consecutive drum cycles, provided that the number of executions of this command, before the format change, is even.

(b) If an odd number of executions of a punch command of the form 01.N.0.10.31 occurs in consecutive drum cycles before a change in format (02.03), then a punch command of either the same form or W.00.N.0.10.31 must be executed in the second drum cycle after the last punch command preceding the format change.

(c) The format (02.03) may be changed between the last two in a sequence of punch commands of the form W.00.N.0.10.31, executed in consecutive drum cycles, provided that the number of executions of this command, before the format change, is odd.

(d) If an even number of executions of a punch command of the form W.00.N.0.10.31 occurs in consecutive drum cycles before a change of format (02.03), then a punch command of either the same form or 01.N.0.10.31 must be executed in the second drum cycle after the last punch command preceding the format change.

(e) The format (02.03) may be changed between the last two in a sequence of punch commands of the form 01.N.0.10.31, executed in alternate drum cycles.

The simplest program giving a change of format after punching leader applies either (b) or (d) above.

Line 19 contains output data; words 02, 01, and 00 of line 02 contain the latter portion of the data format; word 03 of line 02 contains any number whose most significant hex digit is 8 or 9. The leading portion of the data punch format is stored in word 03 of a long or short line or the odd half of a two word register. C_1 and C_2 are any two positive constants whose ratio is approximately twenty times the number of inches of leader to be punched.

Location	P	T	N	C	S	D	Notes
N_0		T_0	N_1	0	S	28	$C_1 \rightarrow AR$
N_1		T_1	N_2	3	S	29	$C_2 \rightarrow AR+$
N_2		T_2	N_3	0	22	31	Sign (AR) → Test
N_3	*	*	N_1	0	10	31	Punch leader
$N_3 + 1$		03	N_4	0	S	02	Data format → 02.03
N_4		01	N_5	0	10	31	Punch (19)

* Use no prefix and T_3 = 01 if the largest integer in the ratio C_1/C_2 is odd; use prefix W and T_3 = 00 if this integer is even.

The punch command in N_3 must be executed [illegible] consecutive drum cycles; therefore $00 < N_1 < T_1 < N_2 < T_2 - 1 < N_3 <$ [illegible] $03 < N_4 < 07$.

ACTION BY: All personnel concerned.

REFERENCES: Technical Applications Memorandum No. 20
Technical Applications Memorandum No. 26

REMARKS:

(1) The techniques described in this memorandum are less cumbersome than the one applied in the program of Technical Applications Memorandum No. 20.

(2) This memorandum does not apply to G-15 output through AN-1.

PREPARED BY: S. H. Lewis

APPROVED BY: T. Yamashita

SHL/TY/mf

TECHNICAL APPLICATIONS MEMORANDUM NO. 39

9 June 1958

TITLE: Precession of Line 19

PURPOSE: A method for programming a four-word precession in Line 19

EQUIPMENT AFFECTED: G-15D

EFFECTIVE DATE: 9 June 1958

INSTRUCTION:

It is generally desirable to minimize the time of an output from Line 19 by a succession of four-word precessions of the data in that line until words u4 - u7 are not all clear. This can be accomplished by a sequence of four commands requiring two drum cycles for each four-word precession. Zeros are introduced into words 00 - 03. The computer must be in the READY state before the following program is executed, so a test for that condition may be necessary.

LOC	P	T	N	C	S	D	Notes
u3	u	00	(L)	0	19	27	(19.u4-u7) → Test
(L)		(L+2)	(M)	0	08	31	=0: Typeout
(L+1)		---	-	-	--	--	≠0: Exit
(M)		(M+2)	99	0	00	31	Set Ready
99	u	u4	(L)	0	19	27	(19.u0-u3) → Test

The commands in locations L and M, which prepare for the precession, must be executed in the same drum cycle not later than time u3. No typeout actually occurs. The test in location 99 must be executed in the following cycle, while the precession is taking place. Therefore, $00 \leq L \leq 96$; $98 \leq M \leq u2$.

If tape is to be punched, words 00 - 03 should not be all clear before the precession. The precession should be followed by one of the punch programs described in Technical Applications Memorandum No. 33.

ACTION BY: All personnel concerned.

REFERENCE: Technical Applications Memorandum No. 33

REMARKS: None

PREPARED BY: S. H. Lewis SHLewis

APPROVED BY: T. Yamashita

SHL/TY/mab

SUPPLEMENT NO. I

To

TECHNICAL APPLICATIONS MEMORANDUM NO. 39

Acknowledgment is made to Mr. Vincent F. Bobrowicz of the Illinois Power Company for his contribution of the supplement for the benefit of the Bendix Computer Division and all the G-15 users.

TITLE: "Block" Precession of Line 19

PURPOSE:
1. To precess faster than normal precession speeds
2. To permit calculation during precession

EQUIPMENT AFFECTED: Bendix G-15D

EFFECTIVE DATE: 27 July 1959

INSTRUCTION:

The data used in several of our programs is in four word blocks. Some of these programs call for precessing these blocks around the drum various multiples of four word times.

Ordinarily, precession of a four word block of data would take four drum cycles plus four word times. Other calculations cannot take place during this time.

Technical Applications Memorandum No. 39 outlines a method for programming a four word precession in line 19; however, the application discussed in Technical Applications Memorandum No. 39 does not provide a recirculating type of precession.

A four command routine has been written which is an extension of the technique outlined in Technical Applications Memorandum No. 39. In programs that are not input-output limited, this routine can be used to process line 19, completely, around the drum in a total of 120 word times for a four word precession.

During this time other calculations can be performed during all but 14 word times of the total 120. Information can be read from line 19 (in unprecessed form) during the drum cycle that precession is taking place.

In the event that more than one block is to be precessed at a time, two additional drum cycles would be required for each additional consecutive four word precession. Calculation can take place during precession as outlined above.

The program to accomplish precession is as follows:

L	P	T	N	C	S	D	
99		u1	u1	0	08	31	Type (AR)
u1		u3	u3	0	00	31	Set Ready
u3	u	00	00	0	19	22	19.u4 - .u7 to L. 22
00							Calculation
			u7				Calculation
u7	u	24	24	0	22	19	L. 22 to 19.00 - .03
04							Next Command

In programming multiple "block" precessions, it is only necessary to loop back to location 99. For an optimized program, the loop should return to location 99 in the same drum cycle as the command in location 04 above. Calculation may take place between 04 and 99.

Vincent F. Bobrowicz
Illinois Power Company

ILLINOIS POWER COMPANY

G-15D CODING SHEET

Problem: Test "Block" Precession

Prepared By: V. F. Bobrowicz

Page 1 of 1

Date: 7/27/59

Line 00

	L	P	T	N	C	S	D	BP	Notes
y	00	u	01	02	0	29	18		Clear L. 18
	02		03	05	1	31	31		Number Track to L. 18
	05	u	06	06	0	18	19		L. 18 to L. 19
	06		08	09	0	29	23		Clear 23.00
	09		11	12	0	12	31		Gate type - in
	12		12	12	0	28	31		Ready?
	13		16	04	1	23	28		(23.00) to AR_c
	04		05	98	0	28	27		Test (AR) for Non-zero
	98		99	u0	0	17	31		Bell Ring
	u0		u2	u2	0	09	31		Type L. 19
	u2		u4	u2	0	16	31		HALT
y	99		u1	u1	0	08	31		Type (AR) *
	u1		u3	u3	0	00	31		Set Ready *
	u3	u	00	01	0	19	22		(19.u4 - .u7) to L. 22 *
	01		50	u7	3	00	29		Decrement (AR)
	u7	u	24	24	0	22	19		(L.22) to 19.00 - .03 *
n	50								0000001

af

1

a y868000	y767000	y666000	y565000
y464000	y363000	y262000	y161000
y060000	xz5z000	xy5y000	xx5x000
xw5w000	xv5v000	xu5u000	x959000
x858000	x757000	x656000	x555000
x454000	x353000	x252000	x151000
x050000	wz4z000	wy4y000	wx4x000
ww4w000	wv4v000	wu4u000	w949000
w848000	w747000	w646000	w545000
w444000	w343000	w242000	w141000
w040000	vz3z000	vy3y000	vx3x000
vw3w000	vv3v000	vu3u000	v939000
v838000	v737000	v636000	v535000
v434000	v333000	v232000	v131000
v030000	uz2z000	uy2y000	ux2x000
uw2w000	uv2v000	uu2u000	u929000
u828000	u727000	u626000	u525000
u424000	u323000	u222000	u121000
u020000	9z1z000	9y1y000	9x1x000
9w1w000	9v1v000	9u1u000	9919000
9818000	9717000	9616000	9515000
9414000	9313000	9212000	9111000
9010000	8z0z000	8y0y000	8x0x000
8w0w000	8v0v000	8u0u000	8909000
8808000	8707000	8606000	8505000
8404000	8303000	8202000	8101000
-1414794	yv6v000	yu6u000	y969000

ef

s-1414794	yv6v000	yu6u000	y969000
y868000	y767000	y666000	y565000
y464000	y363000	y262000	y161000
y060000	xz5z000	xy5y000	xx5x000
xw5w000	xv5v000	xu5u000	x959000
x858000	x757000	x656000	x555000
x454000	x353000	x252000	x151000
x050000	wz4z000	wy4y000	wx4x000
ww4w000	wv4v000	wu4u000	w949000
w848000	w747000	w646000	w545000
w444000	w343000	w242000	w141000
w040000	vz3z000	vy3y000	vx3x000
vw3w000	vv3v000	vu3u000	v939000
v838000	v737000	v636000	v535000
v434000	v333000	v232000	v131000
v030000	uz2z000	uy2y000	ux2x000
uw2w000	uv2v000	uu2u000	u929000
u828000	u727000	u626000	u525000
u424000	u323000	u222000	u121000
u020000	9z1z000	9y1y000	9x1x000
9w1w000	9v1v000	9u1u000	9919000
9818000	9717000	9616000	9515000
9414000	9313000	9212000	9111000
9010000	8z0z000	8y0y000	8x0x000
8w0w000	8v0v000	8u0u000	8909000
8808000	8707000	8606000	8505000
8404000	8303000	8202000	8101000

af

1u

■ 8404000	8303000	8202000	8101000
-1414794	yv6v000	yu6u000	y969000
y868000	y767000	y666000	y565000
y464000	y363000	y262000	y161000
y050000	xz5z000	xy5y000	xx5x000
xw5w000	xv5v000	xu5u000	x959000
x858000	x757000	x656000	x555000
x454000	x353000	x252000	x151000
x050000	wz4z000	wy4y000	wx4x000
ww4w000	wv4v000	wu4u000	w949000
w848000	w747000	w646000	w545000
w444000	w343000	w242000	w141000
w040000	vz3z000	vy3y000	vx3x000
vw3w000	vv3v000	vu3u000	v939000
v838000	v737000	v635000	v535000
v434000	v333000	v232000	v131000
v030000	uz2z000	uy2y000	ux2x000
uw2w000	uv2v000	uu2u000	u929000
u828000	u727000	u626000	u525000
u424000	u323000	u222000	u121000
u020000	9z1z000	9y1y000	9x1x000
9w1w000	9v1v000	9u1u000	9919000
9818000	9717000	9616000	9515000
9414000	9313000	9212000	9111000
9010000	8z0z000	8y0y000	8x0x000
8w0w000	8v0v000	8u0u000	8909000
8808000	8707000	8606000	8505000

ef

2

y464000	y363000	y262000	y161000
y060000	xz5z000	xy5y000	xx5x000
xw5w000	xv5v000	xu5u000	x959000
x858000	x757000	x656000	x555000
x454000	x353000	x252000	x151000
x050000	wz4z000	wy4y000	wx4x000
ww4w000	wv4v000	wu4u000	w949000
w848000	w747000	w646000	w545000
w444000	w343000	w242000	w141000
w040000	vz3z000	vy3y000	vx3x000
vw3w000	vv3v000	vu3u000	v939000
v838000	v737000	v636000	v535000
v434000	v333000	v232000	v131000
v030000	uz2z000	uy2y000	ux2x000
uw2w000	uv2v000	uu2u000	u929000
u828000	u727000	u626000	u525000
u424000	u323000	u222000	u121000
u020000	9z1z000	9y1y000	9x1x000
9w1w000	9v1v000	9u1u000	9919000
9818000	9717000	9616000	9515000
9414000	9313000	9212000	9111000
9010000	8z0z000	8y0y000	8x0x000
8w0w000	8v0v000	8u0u000	8909000
8808000	8707000	8606000	8505000
8404000	8303000	8202000	8101000
-1414794	yv6v000	yu6u000	y969000
y868000	y767000	y666000	y565000

ef

3

8 y060000	xz5z000	xy5y000	xx5x000
xw5w000	xv5v000	xu5u000	x959000
x858000	x757000	x656000	x555000
x454000	x353000	x252000	x151000
x050000	wz4z000	wy4y000	wx4x000
ww4w000	wv4v000	wu4u000	w949000
w848000	w747000	w646000	w545000
w444000	w343000	w242000	w141000
w040000	vz3z000	vy3y000	vx3x000
vw3w000	vv3v000	vu3u000	v939000
v838000	v737000	v636000	v535000
v434000	v333000	v232000	v131000
v030000	uz2z000	uy2y000	ux2x000
uw2w000	uv2v000	uu2u000	u929000
u828000	u727000	u626000	u525000
u424000	u323000	u222000	u121000
u020000	9z1z000	9y1y000	9x1x000
9w1w000	9v1v000	9u1u000	9919000
9818000	9717000	9616000	9515000
9414000	9313000	9212000	9111000
9010000	8z0z000	8y0y000	8x0x000
8w0w000	8v0v000	8u0u000	8909000
8808000	8707000	8606000	8505000
8404000	8303000	8202000	8101000
-1414794	yv6v000	yu6u000	y969000
y868000	y767000	y666000	y565000
y464000	y363000	y262000	y161000

TECHNICAL APPLICATIONS MEMORANDUM NO. 40

11 July 1958

TITLE: The use of the Metalised Version of the Mylar Tape.

PURPOSE: To prevent accidents

EQUIPMENT AFFECTED: G-15D

EFFECTIVE DATE: 11 July 1958

INSTRUCTIONS:

Recently experiments have been conducted in the use of the metalized version of the Mylar Tape on the G-15 computer. It was found that because of the conductive material on the tape, the Mylar Tape is very dangerous to use. Not only serious damage to the machine could be done, but also it represents a danger to human lives.

It is strongly recommended that the metalized version of the Mylar Tape <u>not</u> be used on the G-15D Computer.

ACTION BY: All personnel concerned

REFERENCES: None

REMARKS: None

PREPARED BY: T. Yamashita
APPROVED BY: T. Yamashita

TECHNICAL APPLICATIONS MEMORANDUM NO. 41

TITLE: Special uses of the "Select command line and return to mark place" Command. (L_k N C 20 31)

PURPOSE:

1. To demonstrate a method of programming option selection which depends only on the position of the compute switch.

2. To elaborate on the use of the L_k N C 20 31 as an unconditional transfer command.

EQUIPMENT AFFECTED: G-15D

EFFECTIVE DATE: August 8, 1958

INSTRUCTION:

(1) The program can choose a loop depending on the position of the compute switch (Go or BP) through the use of the breakpointed "return to mark place" command (L_k N C 20 31). If this command is written with a "B.P." and if it is run at B.P. it will always return to the mark regardless of the N number. If the T is written equal to N then in normal "GO" operation the program will go to N and not to the mark. It should be noted that the computer will not halt on a B.P. return to mark place command (L_k N C 20 31-) but will continue to the next command. (Refer to Technical Memorandum #4). It will halt here.

An illustration of this follows: In "GO" operation typing is gated, in B.P. typing is not gated.

L		T	N	C	S	D	
00	w	05	01	6	21	31	Mark place
01	u	02	02	0	19	00	L 19 ⟶ L00
02		09	09	0	20	31 -	Return to mark place
09		11	05	0	12	31	Gate typing
05		05	00	0	16	31	HALT

(2) The "Return to Mark Place" command can be used as an unconditional transfer regardless of where the mark is if it is written in the following form:

N N C 20 31

This will enable transferring without losing the existing mark or making a new one. This is a modification of the L_k N C 20 31 command referred to in Applications Technical Memorandum No. 4. To achieve the unconditional transfer effect the L_k should always be set equal to N as the computer does not start looking for a mark until the L_k time. An illustration of this method follows. Although the mark is at T = .05 and the L of the return to mark place is at =.02 the program always gates typing. (It doesn't go to the mark place but to the N).

L		T	N	C	S	D	
00	w	05	01	6	21	31	Mark
01	u	02	02	0	19	00	L19 → L00
02		06	06	0	20	31	Return to mark
06		08	05	0	12	31	Gate typing
05		05	00	0	16	31	HALT

Action by: All personnel concerned.

References: Technical Memorandum No. 4 dated 11 April 1957.

Prepared by: Basil Mikhalkin

Approved by: T. Yamashita

Supplement No. 1

TECHNICAL APPLICATIONS MEMORANDUM # 41

Title: Additional Uses of the "Select Command Line and Return to Mark Place" Command (L_k N C 20 31).

Purpose:
1. To demonstrate the use of this command as an automatic loop counter.
2. To demonstrate the use of this command as a many-way switch.

Equipment Affected: G15D.

Effective Date: 2 January 1959

Instructions

A program may be required to execute a loop n times, after which it will exit automatically to the marked place. The technique to accomplish this is to increment L_k each time through the loop.

```
       T  N  C  S  D

00     01 02 0 01 28             01:01 - 12 19 1 20 31
02 W   53 19 1 21 31


19     20 21 0 28 22

       - - - - - - - -
       - - - - - - - -
       - - - - - - - -
05     06 07 1 01 28             1 x 2^-8
07     08 09 1 22 29
09     11 11 0 31 31             N C A R
```

This program will return to location N-19 until L_k-13, 14, - - reaches 20; at which time the command will search for the mark - 53.

$n - N - L_k$

2. A program may be required to exit after completing a number of distinct loops. To accomplish this we increment (or decrement) N.

```
       T  N  C  S  D

00     01 02 0 01 28             01:01 - 15 19 1 20 31
02     04 05 0 28 22
05 W   53 19 1 21 31
```

15		- Loop 5
16		- Loop 4
17		- Loop 3
18		- Loop 2
19		- Loop 1

\- - - - - - -
\- - - - - - -
\- - - - - - -

C3	04 06 1 22 28	
C6	07 08 3 01 29	Subtract 1×2^{-16}
08	12 13 1 28 22	
13	15 15 0 31 31	N C A R

This program will choose successive loops 1 through 5 dependent upon the N position of the command.

3\. This technique may also be used to exit upon the convergence of a series by adding an increment to the looping instruction if and only if the series has passed a test for convergence.

4\. This is probably the most powerful looping instruction in the computing field today.

John C. Laffan, Jr.

Scientific Design Co

TECHNICAL APPLICATIONS MEMORANDUM NO. 42

28 July 1958

TITLE: Standard Program Preparation Routine - Correction of an Error in Decimal to Binary Conversion Routine

PURPOSE: To avoid loss of information in L.03.u4 and L.03.u5 when using the "v" instruction

EQUIPMENT AFFECTED: G-15D

EFFECTIVE DATE: 28 July 1958

INSTRUCTION:

Attention has been called to the fact that using the "v" instruction in the PPR magazine causes word positions 03.u4 and 03.u5 to be cleared.

To correct existing PPR tapes, execute the following steps:

1. Place PPR magazine on photo-reader and read in the basic package (Lines 15, 16, 17, 05).

2. Type "w00 (tab) s" and place the compute switch to break-point repeatedly until the check sum "-7w0z6u2" is typed.

3. With the compute switch off, strike the f key with enable on, then place the compute switch to GO.

4. Type "zu2 (tab) -y802357 (tab) s". The type-back should be "-y802343".

5. Type "z90 (tab) 0000014 (tab) s".

6. Type "x06 (tab) s" and punch the corrected line. The check sum will be "-7w0z6u2".

7. Replace the incorrect line in the magazine with the correct one (after first rereading to insure punching is correct).

ACTION BY: All personnel concerned

REFERENCES: G-15D Operating Manual
Technical Applications Memorandum No. 38

REMARKS: None

PREPARED BY: E. S. Williams *E. S. Williams*
APPROVED BY: T. Yamashita

EW/TY/bg

TECHNICAL APPLICATIONS MEMORANDUM NO. 44

TITLE: Miscellaneous optional modifications to the Intercom 1000 system, dated August 24, 1958 and January 14, 1959.

NOTE: This memo is not applicable for Intercom 1000 tapes dated other than noted.

PURPOSE: 1. To provide advanced preliminary information on optional changes that can be made to the Intercom 1000 system.

a. Elimination of input verifications - Intercom 1000 double precision.

b. Elimination of input verifications - Intercom 1000 single precision.

c. The desk calculator mode.

d. The storing of Intercom 1000 double precision using the Magnetic Tape Service Routine.

EQUIPMENT AFFECTED: G-15D

EFFECTIVE DATE: 1 December 1959

INSTRUCTION:

a. ELIMINATION OF INPUT VERIFICATIONS - INTERCOM 1000 DOUBLE PRECISION.

The following changes to the basic INTERCOM Package make it possible to eliminate the type-back of input.

To eliminate the type back of

A. Commands

	Computer	Operator *	Computer
1.	-	500816	C. R.
2.	816	1w39752	1w39752 C. R.
3.	817	0670000/	Ring Bell

B. Numbers

	Computer	Operator *	Computer
1.	-	500282	C. R.
2.	282	-x833645	-x833645 C. R.
3.	283	0500727/	C. R.
4.	727	682vz86	682vz86 C. R.
5.	728	0670000/	Ring Bell

* Each of the operations performed by the operator must be followed by a "tab s".

b. ELIMINATION OF INPUT VERIFICATIONS - INTERCOM 1000 SINGLE PRECISION.

The commands listed under "A" must be placed in Intercom for the elimination of input verification of numbers both fixed and floating. The commands under "B" must be placed in Intercom to eliminate the typeback of commands during input.

A.	02.74	u5.	u4.	5.	20.31	
	05.u4	00.	80.	0.	21.	28
B.	06.u1	w.30.u4.5.21.31				
	05.u4	00.	80.	0.	21.	28

The above commands are expressed in FPR language. To use Intercom as a means of input for these commands they must be expressed in sexadecimal. The following is a sexadecimal representation of the commands above:

A.	02.74	689z96-
	05.u4	802vw05
B.	06.ul	986vzy6-
	05.u4	802vw05

Since the command input routine for Intercom doesn't accept negative numbers, it is necessary to clear and subtract the numbers in 02.74 and 06.ul and store these numbers back with the minus sign attached. Please note the enclosed sheet for example.

c. THE DESK CALCULATOR MODE.

When in the manual mode of operation, INTERCOM may be treated as a desk calculator -- that is, when any arithmetic command is executed, the result of its execution will be typed out in fixed-point, followed by a tab. Before operating INTERCOM as a desk calculator, perform the operations listed under A. Before attempting to execute a program in the automatic mode, perform the operations listed under B.

A.

	Computer	Operator *	Computer
1.	---	500871	C. R.
2.	871	386vz83	386vz83 C. R.
3.	872	0500136/	C. R.
4.	136	-u12v942	-u12v942 C. R.
5.	137	0670000/	Ring Bell

B.

	Computer	Operator *	Computer
1.	---	500871	C. R.
2.	871	152vz51	152vz51 C. R.
3.	872	0670000/	Ring Bell

* Each of the operations performed by the operator must be followed by a "tab s".

d. THE STORING OF INTERCOM 1000 DOUBLE PRECISION USING THE MAGNETIC TAPE SERVICE ROUTINE.

Assuming that the computer is on, the Magnetic Tape Unit is on and INTERCOM 1000 D.P. is in memory, four constants must be stored in Line 05 before INTERCOM 1000 D.P. can be written on magnetic tape.

Store the following constants in Line 05, words 00, 01, 04 and 05, respectively, using the INTERCOM input mode initiated by op code 50. (Type in commands).

```
810v440-
002vz00
zzz00zz-
zzzzzzz-
```

Write on magnetic tape the lines 00, 01, 02, 03, 04, 06, 07 and 08 followed by line 05. Instruct the "Magnetic Tape Service Routine" that the last line to be written is line 05 and that execution is to begin with word 00 of line 05.

ACTION BY: All personnel concerned

REFERENCES: Applications Project No. 61
Magnetic Tape Service Routine

PREPARED BY: Don Hassell

APPROVED BY: T. Yamashita