(Alphanumeric Output Subroutine continued)

4-BIT ALPHANUMERIC OUTPUT CODES

) 0	04		Aa	72
L 1	OJ		Bb	OF
@ 2	14		Cc	6F
" 3	1J		Dd	2F
$ 4	24		Ee	4F
% 5	2J		Ff	54
* 6	34		Gg	5J
π 7	3J		Hh	62
Σ 8	44		Ii	22
(9	4J		Jj	64
Space	56	X	Kk	6J
_ -	OA		Ll	OJ
△ ∈	16		Mm	3F
: ;	1A		Nn	32
? /	26		Oo	46
] .	2A		Pp	42
[,	36		Qq	74
TAB	30		Rr	1F
Lower Case	08		Ss	7F
Upper Case	10		Tt	5F
Color Shift	18		Uu	52
Carr. Ret.	20		Vv	3A
Back Space	28		Ww	7J
= +	06	X	Xx	4A
Stop Code	40		Yy	12
Leave Routine	VQ		Zz	02

Range :

X = different for 6 bit

- CONTENTS -

LGP-30 SUBROUTINE MANUAL

This subroutine manual was compiled by the Royal McBee Computing Section to facilitate coding problems for the LGP-30. The programming was done by the Royal McBee Computing Section and various LGP-30 users. As other subroutines are developed they will be added to the manual and distributed to LGP-30 users.

The subroutines provide means for machine input and output as well as evaluations of basic functions which are used in many problems. For each subroutine the program listing, calling sequence, running time, and storage requirements are given. Also, where applicable, the range of variables, scaling, and accuracy are given. The values of constants used in the subroutine are given in the program listing in the "note" column of the line on which the constant is represented.

It is hoped that as LGP-30 users develope other useful subroutines they will submit the routines to the Royal McBee Computing Section for inclusion in this manual. This will facilitate prompt and complete distribution of generalized routines and prevent unnecessary duplication.

Royal McBee Corporation
1560 North La Brea Avenue
Hollywood 28, California

Copy No. 237

CONVENTIONS USED IN THESE WRITE -UPS AND LISTINGS

1. α is the base memory location from which entry to a subroutine is executed. Locations used by the subroutine are in reference to location α. E.G., $\alpha + 1$, $\alpha + 2$, $\alpha + 3$......

2. Memory locations within the subroutine which are not used are left blank. Those used as temporary storage are bracketed. If the address of an instruction is modified in the execution of the subroutine, the address is bracketed.

3. We try to use the "Hex. words" input code as infrequently as possible. However, when this code is used, we try to load as many words as possible with it.

4. L_o designates an initial location. L_f designates the final location.

5. It should be the programmer's responsibility to enter "Program Input Codes" (and the associated stop code) on the coding sheet. This will usually consist of a start fill (;), a set modifier (/), and some hex. words (,) codes.

6. The "Stop" and "Stop Codes" referred to in these write-ups and on the coding sheet are synonymous with "Conditional Stop Code".

7. For explanation of our scaling convention, see write-up on "SCALING".

8. Track 63 is used by some of these subroutines for temporary storage. The track 63 sectors used by the subroutines are enumerated in the respective write-ups. This practice was found useful for "optimum" programming of subroutines. However if the subroutines which use this temporary storage are to remain optimum, the L_o of the subroutine must be the beginning of a track. It is suggested that the programmer may also use track 63 for temporary storage of intermediate calculations. He should not place a number in a track 63 location used by one of these subroutines and expect that number to be there after exit from the subroutine.

10/29/56

DEFINITIONS

1. A routine is a logical subdivision of a program, complete in itself, and serving a specific function in the problem. There is no fixed length to any routine, and each routine occupies only as much storage as is actually needed.

2. A subroutine consists of a set of instructions to perform a standard task which is of a sufficiently general nature to be used in a number of different programs. Examples are subroutines to input and output data, compute square roots, arctangents, etc. This Subroutine Manual is a compilation of the specifications and listings of the subroutines, completely describing the function and use of each.

3. A calling sequence is a set of instructions used for transferring from the main routine to a particular subroutine. It may also include information needed by the subroutine, such as constants and the locations of certain quantities. The calling sequence for each of the subroutines is given in the Subroutine Manual.

4. Minimum Time Programs

There are occasions when it is necessary to write programs which will be executed in as little time as possible. These minimum time programs are referred to as "optimum" programs. Since the subroutines contained in the Manual are to be used over and over again, they have been optimized. (The process of optimizing requires placing the sector of the operand α at $\alpha + (7k+1)$ where $2 \leq k \leq 6$ for most instructions).
The programmer should bear in mind that 10,000 executions of all non-optimum instructions would take less that 3 minutes longer than 10,000 executions of optimum instructions. If the programmer spends 15 - 30 minutes on each routine trying to save machine time by optimizing, this time may never be made up in the actual running of the problem.

SCALING

The LGP-30 normally handles all numbers as if they were of the form .XXXX.........., that is, numbers numerically less than 1. However, it is quite simple to carry any number in the machine at any number of binary places, and this arithmetic is explained below. In talking about the placement of the radix point in the LGP-30, it is simpler to talk of the number of whole places in front of the radix point, rather than the number of places after the point. Hereafter, a number will be referred to as being carried at q places, q being the number of binary digits to the left of the radix point, and 30-q as the number to the right of the point.

Addition: Addition of course poses no problem if the two numbers to be added are at the same number of places. If not, either may be shifted before addition by multiplying or dividing by "One" at an appropriate q.

Multiplication: The LGP-30 multiplies a number at q_1 places by a number at q_2 places and forms the product in the accumulator at q_1 plus q_2 places.

in mult. only in binary pt.

Division: The LGP-30 divides the accumulator at q_1 places by a number at q_2 places and forms the quotient in the accumulator at $q_1 - q_2 = q_3$ places. It should be noted that overflow will occur if the quotient developed is not less than 2^{q_3} in absolute value.

PUNCHING TAPES FROM CODING SHEETS

1. Only the "Program Input Codes" and "Instruction" columns of the coding sheet are to punched, with appropriate stops. Never punch "Location," "Contents of Address," or "Notes" columns.

2. Each entry on a line must be followed by a conditional stop code--"Stop" column, symbol ('). A line left blank must have the stop code punched.

3. Punch the "Program Input Codes" column only when there is an entry in the column. The "Program Input Codes" must be followed by the stop ('). This punching must precede the punching of the "Instruction" column on the same line of the coding sheet.

4. Leading zeros need not be punched. All other zeros must be punched. E.G., 00013086' only 13086' need be punched. ,0000017' must be punched ,0000017'. For T0059' punch T0059'.

5. Consider brackets as containing zeros. E.G., for [, , , ! , , ,]' = [00000000] ', only the stop code need be punched. For B[, , ,]' = B[0000]' punch B0000'.

6. All punching may be done in lower case. B0627' will appear as b0627'.

7. The placing of carriage returns is left to the discretion of the person preparing the tape. Carriage returns do not affect the input operation. We have arbitrarily placed a carriage return (⌧) after every 4 words on each coding sheet.

8. A heading may precede a punched program to identify the tape. Anything except a stop code may be punched as a header. Then as the tape is fed through the input reader the heading will print but will not affect the operation of the computer.

9. Each tape should be verified after punching. This can be done by placing the punched tape in the reader and "listing" the tape by the following process.

   a. Make sure "Cond. Stop" button on reader is up.
   b. Depress "Start Read" button.
   c. When printing stops, depress space bar.
   d. Repeat steps b and c until entire tape is printed.

Then the printing may be visually checked against the coding sheets for correctness and presence of stop codes.

NOTE

This subroutine manual has been written for use with the 6-bit Flexowriter. This Flexowriter is standard equipment on all LGP-30's with serial numbers above 12. Certain routines contained herein must be modified if they are to be used on the earlier model 4-bit Flexowriter. These modifications are listed on the "Changes for 4-bit Flexowriter" page.

The following keyboard codes differ in the 4-bit and 6-bit Flexowriters:

4-BIT	CODES	6-BIT
	123456	
@ 2	001010	* 2
$ 4	010010	Δ 4
* 6	011010	$ 6
' +	000011	Space
Space	101011	Not recognized
Δ ϵ	001011	= +
Does not print	100000	'

4/3/57

CHANGES FOR 4 - BIT FLEXOWRITER

1. Program 10.0 write-up.
   Change "+" to "€" (three changes).

2. Program 12.0 coding
   Location 0024. Change from 03F0 to 2GF0.

3. Program 13.0 coding.
   Location 0056. Change from XP0314 to XP4314

4. Program 13.1 coding.
   Location 0056. Change from XP0314 th XP4314

5. Program 20.0 output codes.
   Substitute "4-bit Alphanumeric output codes" sheet for " 6-bit Alphanumeric output codes" sheet.

6. Program 21.0 coding.

Location	0008.	Change	from	XP0344	to	XP4344
"	0034	"	"	XP0306	"	XP4306
"	0107	"	"	XP0336	"	XP4336
"	0218	"	"	XP1154	"	XP0354

7. Program 12.1 coding.
   Location 0204. Change from 803J0 to 82GJ0

3/28/57

ROYAL McBEE CORPORATION

LGP-30 Input Output

Keyboard Code

Numerical

	123456
)0	000010
Ll	000110
*2	001010
"3	001110
Δ4	010010
%5	010110
$6	011010
π7	011110
Σ8	100010
(9	100110
Ff	101010
Gg	101110
Jj	110010
Kk	110110
Qq	111010
Ww	111110

Commands

	123456
Zz	000001
Bb	000101
Yy	001001
Rr	001101
Ii	010001
Dd	010101
Nn	011001
Mm	011101
Pp	100001
Ee	100101
Uu	101001
Tt	101101
Hh	110001
Cc	110101
Aa	111001
Ss	111101

Controls

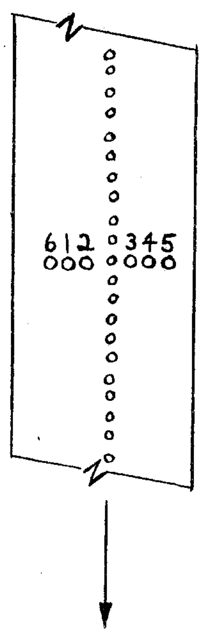
	123456	
Lower Case	000100	
Upper Case	001000	
Color Shift	001100	
Car Ret	010000	
Back Space	010100	
Tab	011000	
Cond Stop (')	100000	
Start Read	000000	
~~Space~~	~~000011~~	OK
Delete	111111	

Signs

~~= +~~	~~001011~~	OK
_ -	000111	

Balance of Keyboard

	123456
:;	001111
?/	010011
].	010111
[,	011011
Vv	011111
Oo	100011
Xx	100111



Note - we have a 6 bit flex!

BOOTSTRAP ROUTINE
(PROGRAM 09.0)

FUNCTION:

To load the input routine on tracks 00, 01 and 02. After the bootstrap program has loaded the entire input routine, a halt is executed at track 63 sector 13 (3W34.) Depressing the Start button transfers control to the first instruction of the input routine.

PROCEDURE:

The tape containing the bootstrap (and the program input routine) is placed in the tape reader and then the following manual operations are performed

1. Connect Switch to "off" position.
("Inp. - Comp." switch to "Comp." on early machines.)

   a. Depress Flexowriter "Start Read" button.

2. Depress "Manual Input" button on console.
("Interrogate" button on early machines.)

3. Depress Flexowriter "Start Read" button.

4. Depress "Fill Instruction" button.
("Fill R" on early machines.)

5. Depress Flexowriter "Start Read" button.

6. Depress "One Operation" button.

7. Depress "Execute Instruction" button.
("Execute R" button on early machines.)

8. Repeat steps 2 through 7 five more times before proceeding to step 9.

9. Depress "One Operation" button.

10. Depress "Normal button.

11. "Connect" switch to "On".
(Inp - Comp." switch to "Inp." on early machines.

12. Depress Computer "Start".

The entire tape will automatiaclly read in after manually performing step 12 above.

LGP-30 PROGRAM INPUT ROUTINE
(PROGRAM 10.0)

The purpose of this report is to describe a method of entering information into the LGP-30. The general characteristics of the LGP-30 are described in the programming manual. A program consists of two types of words, data and instruction. This write-up primarily describes the process of inputing instruction words and hexadecimal representations of data words. This process does not handle data words expressed in decimal form.

There are several functions to be performed by a useful input routine.

1. The most direct way of entering information into the LGP-30 is to present it with binary words. But since it is difficult to program in this number system, we prefer to do our programming in decimal notation. If we are to write words in decimal form, we must provide the machine with a means of converting such words into binary form.

2. Most routines contain instructions which refer to other locations within that routine. Hence if we wish to place the routine in another portion of memory, we must modify some of these addresses.

3. It is sometimes useful to express a number in binary form. e.g., π or other universal constants.

4. It may be necessary to make instructional or data changes to a program that has already been stored in memory.

These are the functions which this input routine is designed to perform.

This routine recognizes seven types of input word. The sign and first 3 bits of the input word are used for the input routine to identify the type. These words and their symbols are as follows:

1. Instruction (none) consists of an order and decimal address. The address consists of a decimal track and sector. The instruction is converted to its binary equivalent and stored in a given location. The address portion is incremented by the contents of the "modifier" (to be discussed below) unless an "x" precedes the order. e.g. b 4000 will in incremented. x b 6310 will not be incremented (and the x will not appear in the stored instruction).

2. Command (+) This word will be treated as an order to the input routine. The order will be executed after entry of another word. The command is input in decimal and is not incremented by the modifier. The second word, presumably data, is input in hexadecimal. e.g. +00h1637 followed by 73W08 will store the hexadecimal word 73W08 in memory location 1637.

PROGRAM 10.0

3. Start fill (;). Tells the input routine where to begin filling input words. Each succeeding word will be filled consecutively. The address portion of the start fill word is decimal, and consists of both track and sector number. e.g. ;0003128. The first stored word will be located in track 31, sector 28.

4. Set modifier (/). The magnitude of the address of the modifier will be used to increment orders. The set modifier word will usually follow a start fill and will usually be identical to it in magnitude. This word is for use by the input routine only.

5. Stop and transfer (.). This word stops the flexowriter. A "start" will transfer control to the memory location contained in the address portion of the stop and transfer word. Depressing break switch 32 on the control panel will cause the computer to disregard the stop portion of this word. e.g. .0001700 will stop reading, then control will be transferred to track 17, sector 00.

6. Hex. words (,). This instruction causes the next N words to be filled without conversion. N is specified in the address portion of the "hex. words" word and must be within the range $1 \leq N \leq 63$ e.g. ,0000014 means the next 14 words are to be stored in the next 14 consecutive memory locations. The words must be in hexadecimal notation, and they will not be incremented by the modifier.

7. Hex. fill (v). Fills the next n words hexadecimally beginning in m. m and n are proper hexadecimal numbers. The format of the word is v n_1 n_2 n_3 m_1 m_2 m_3 m_4. e.g. v1J02W00 means the next 1J0 words $[(1J0)_{16} = (448)_{10}]$ will be filled consecutively beginning in location 2W00 $[(2W00)_{16}$ = track $(47)_{10}$ sector 00]. Up to $(7WW)_{16} = (2047)_{10}$ words can be filled by a single hex fill input order.

Leading zeros need not be punched on any input word. All other zeros must be punched. e.g. 800T0018 must be completely punched; 000B3749 only the last five characters need be punched. e.g. B3749.

When the overall coding for a problem is surveyed, it is found that the instructions separate logically into independent groups, some of which can be used in any number of problems. Examples of these groups are subroutines of all types, standard input and output routines, and the mathematical subdivisions of the problem. It would be desirable to code these pieces without reference to the other pieces. In order to separate these pieces completely, it is necessary to assign a group of instructions a block of storage locations which does not correspond to actual memory locations; otherwise two blocks of coding might be found to occupy the same section of storage, requiring a change in the

PROGRAM 10.0

coding for one of the two pieces. The "set Modifier" order of the input routine was intended to facilitate this type of coding. A group of instructions can be coded without reference to actual memory locations by starting that group at symbolic address 0000. Then by setting the modifier to the "start fill" location, the programmer may position a routine to any part of memory. An instruction preceeded by an "x" will not be incremented, and this instruction will still refer to an absolute memory location. It should be noted that orders may be coded for actual locations merely by setting the modifier to zero (i.e., input order /0000000). Thus no particular restrictions are imposed upon the programmer by this system.

If the input routine detects an erroneous input code it will print "code" and halt. The last word read from tape contains the erroneous code in the first punched character.

A tape prepared for this input routine must contain flexowriter format control. It is suggested that a carriage return be inserted after every four words on tape. If there is no format control and the flexowriter carriage is permitted to space into the automatic carriage return a stop will result. The computer will continue if the carriage return button is depressed.

INPUT ROUTINE FORMAT
PROGRAM 10.0

Code / Operation / Track / Sector		INTERPRETED AS:
Op 1 2 3 4	Mod.	Instruction - Order plus address (dec.)
x Op 1 2 3 4	Not Mod.	
- 8 0 0 T 1 2 3 4	Mod.	
- 8 0 x T 1 2 3 4	Not Mod.	
+ 0 0 Op 1 2 3 4		Command. This word is treated as an order, using the following word as data. Following word in hex.
; 0 0 0 1 2 3 4		Start Fill - Address in decimal
/ 0 0 0 1 2 3 4		Set Modifier - Address in decimal
. 0 0 0 1 2 3 4		Cond. Stop and Transfer - Stop (unless Sw 32 down) and transfer to location specified in address.
, 0 0 0 0 0 1 2		Hexadecimal words. Next N (dec) words are hex. fill sequentially $1 \leqq N \leqq 63$
v $n_1n_2n_3$ $m_1m_2m_3m_4$		Hexadecimal Fill - Fill N (hex) words beginning in loc. M (hex) $1 \leqq N \leqq 2047$

LGP-30 CODING SHEET

Job No. ______ Program No. 10.0 Prepared by M. Kaye Date 10/12/56

Problem Program Input Routine Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
;0000	'					
/0000	'	☒			c0000	
		0000	C0143	'	C01FJ	clear acc
		01	xP0000	'		Input one word
		02	xI0000	'		
		03	T0121	'	☒	→ Neg test order or less
		04	S0054	'	1@3	
		05	T0034	'		→ Order
		06	S0156	'	1@3	
		07	T0245	'	☒	→ Spec. Code
		08	S0144	'	1@3	
		09	T0149	'		→ Command for
		10	U0011	'		Input routine
		11	S0054	'	☒ 1@3	
		12	T0213	'		→ Start fill
		13	S0156	'	1@3	
		14	T0220	'		→ Set modifier
		15	S0144	'	☒ 1@3	
		16	T0227	'		→ Halt + transfer
		17	U0018	'		
		18	S0054	'	1@3	
		19	T0114	'	☒	→ N hex. words
		20	H0163	'		here for hex. fill
		21	E0207	'	07WW0000	extract location
		22	N0244	'	1@27	
		23	U0026	'	☒	
;00000001	'	24	K00000000	'	-6@4 (0052)	
		25		'		
		26	H0162	'	$n_1 n_2 n_3$ (hex)	
		27	B0163	'	☒ Location (hex)	
		28	U0239	'		
;00000001	'	29	3WWWWJ	'	(0150)	
		30	xP0000	'		
		31	xI0000	'	☒	

' Conditional Stop Code ☒ Carriage Return

LGP-30 CODING SHEET

Job No. ______ Program No. 10.0 Prepared by M. Kaye Date 10/12/56

Problem PROGRAM INPUT ROUTINE Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	☒				
		0032	[ ]	'		execute command
		33	U 0000	'		
		34	A 0156	'	1@3	here for order
		35	N 0200	'	☒ 1@29	
		36	R 0101	'		Binarize routine
		37	U 0048	'		
		38	S 0210	'	1@11(or 16@15)	U019J
		39	T 0042	'	☒	→ Add modifier 0
		40	S 0126	'	14@15	here don't modify 0
		41	U 0139	'		- A020J
		42	A 0128	'	1@11(or 16@15)	here to modify order
		43	U 0138	'	☒	
2000 0004	'	44	F000 0000	'	-3/4@0 (0058)	
		45	WWWWJ	'	1@11 - 1@29 (0059)	
		46		'		
		47	[ ]	'	☒ counter (0102)(0111)	
		48	H 0127	'	N_1	Begin binarization
		49	U 0051	'		
		50	XZ 0001	'	1@29 (0107)(0114)	
		51	E 0202	'	☒ 3J3J3J0	*
		52	M 0024	'	-6@4	*
		53	U 0055	'		
2000 0001	'	54	1000 0000	'	1@3 (0004)(0011)(0018)	
		55	A 0127	'	☒ N_1	*
		56	H 0235	'	N_2	*
		57	E 0208	'	0wwwwww00	*
		58	M 0044	'	-3/4@0	*
		59	U 0063	'	☒	
2000 0001	'	60	4	'	1@29 (0145)	
		61		'		"
		62		'		
		63	A 0235	'	☒ N_2	*

' Conditional Stop Code ☒ Carriage Return

LGP-30 CODING SHEET

Job No. ______ Program No. 10.0 Prepared by M. Kaye Date 10/12/56

Problem PROGRAM INPUT ROUTINE Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	☒				
		0100	U 0101	'		
		01	U [ ]	'	exit from binarize routine	
		02	C 0047	'	counter	
		03	xP 0000	'	☒	Read hex word
		04	xI 0000	'		
		05	C [ ]	'		
		06	B 0139	'		Advance "fill" position
		07	A 0050	'	☒ 1@29	
		08	Y 0105	'	Loc. of fill	
		09	U 0110	'		
		10	Y 0139	'	Loc. of fill	
		11	S 0047	'	☒ counter	Finished?
		12	T 0237	'		→ Not finished
		13	U 0000	'		Finished
		14	N 0050	'	1@29	here for N hex. words
		15	E 0201	'	☒ 0003wwww	
		16	R 0101	'		Binarize routine
		17	U 0048	'		
		18	~~U 0119~~	'		
		19	A 0105	'	☒ 1st location of "hex word" fill	
		20	U 0102	'		
		21	N 0200	'	1@29	here for neg. test order or less
		22	T 0245	'		
		23	S 0209	'	☒ 1@1	
		24	T 0211	'		→ Neg test order
		25	U 0245	'		
2000 0003	'	26	4F0000	'	74@15 (0040)	
		27	]	'	☒ temp (0048)(0055)	
		28	100000	'		
		29	R 0101	'		Binarize routine
		30	U 0048	'		
		31	~~S 0210~~	'	☒ ~~1@11~~	

' Conditional Stop Code ☒ Carriage Return

LGP-30 CODING SHEET

Job No. ______ Program No. 10.0 Prepared by M. Kaye Date 10/12/56

Problem PROGRAM INPUT ROUTINE Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	☒				
		0132	~~T 0136~~	'		→ modify
		33	~~S 0219~~	'	7/@15	here don't modify neg. test order
		34	A 0206	'	80000000	
		35	~~U 0139~~	'	☒	
		36	E 0158	'	800w3wwJ	here modify neg. test order
		37	U 0138	'		
		38	~~A 0203~~	'	~~modifier~~	139
		39	C [ ]	'	☒ ~	
		40	B 0105	'	Loc. of fill	Advance "fill" position
		41	U 0145	'		
1000 0003	'	42	4	'	1@29 (0213)(0220)(0227)	
		43	[ ]	'	☒ temp (0000)	
		44	1000 0000	'	1@3 (0008)(0015)	
		45	A 0060	'	1@29	
		46	Y 0139	'	Loc. of fill	
		47	Y 0105	'	☒ Loc. of fill	
		48	U 0000	'		
		49	N 0200	'	1@29	here for input routine command
		50	E 0029	'	003wwwwJ	
		51	R 0101	'	☒	} Binarize routine
		52	U 0048	'		
		53	C 0032	'	Command + address	
		54	U 0030	'		
		55		'	☒	
1000 0019	'	56	1000 0000	'	1@3 (0006)(0013)(0034)	
		57	3wwwJ	'	(0214)(0221)(0228)	
		58	800w wwwJ	'	(0136)	
		59	0000 0000	'	☒ (0137)	
		60		'		
		61	1000 0000	'	1@1 (0211)	
		62	[ ]	'	temp (0016)(0240)	
		63	[ ]	'	☒ temp (0020)(0027)	

' Conditional Stop Code ☒ Carriage Return

LGP-30 CODING SHEET

Job No. ______ Program No. 10.0 Prepared by [illegible] Date 10/12/56

Problem PROGRAM INPUT ROUTINE Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	☒				
		0200	4	'	1@29 (0035)(0121)	
		01	3wwwJ	'	(0115)	
		02	3J3J3J0	'	(0051)	
		03	[ ]	'	☒ modifier	
		04	wwwJ	'	(0239)	
		05		'		
		06	80000000	'	-1@0 (0134)	
		07	7ww0000	'	☒ (0021)	
		08	wwwwww00	'	(0057)	
		09	40000000	'	1@1 (0123)	
		10	100000	'	1@11 (0038)(0131)	
		11	A0161	'	☒ 1@1	
		12	U0129	'		
		13	N0142	'	1@29	here for "start fill"
		14	E0157	'	0003wwwJ	
		15	R0101	'	☒	Binarize routine
		16	U0048	'		
		17	U0146	'		
		18		'		
90000001	'	19	4F0000	'	☒ 74@15 (0133)	
		20	N0142	'	1@29	here for "set modifier"
		21	E0157	'	0003wwwJ	
		22	R0101	'		Binarize routine
		23	U0048	'	☒	
		24	H0203	'	modifier	
		25	U0000	'		
		26		'		
		27	N0142	'	☒ 1@29	here for halt + transfer
		28	E0157	'	0003wwwJ	
		29	R0101	'		Binarize routine
		30	U0048	'		
		31	Y0233	'	☒	

' Conditional Stop Code ☒ Carriage Return

LGP-30 CODING SHEET

Job No. ______ Program No. 10.0 Prepared by M. Kaye Date 10/12/56

Problem Program Input Routine Track ______

Program Input Codes	Stop	Location	Instruction Op.	Address	Stop	Contents of Address	Notes
	'						
	'	☒					
		0232	xZ	3200	'	Conditional Stop (sw #32)	
		33	U	[ ]	'		
		34			'		
		35	[	]	'	☒ temp (0056)(0063)	
		36			'		
		37	B	0159	'	zero	clear accumulator
		38	U	0103	'		
		39	E	0204	'	☒ 0000wwww	drop ½ $n_1 n_2 n_3$
		40	A	0162	'	$n_1 n_2 n_3$	
		41	S	0263	'	3@15	form: $n_1 n_2 n_3 C\ m_1 m_2 m_3 m_4$
		42	C	0257	'		
		43	U	0255	'	☒	
		44	xZ	0004	'	1@27 (0022)	
		45	xP	5300	'		Print "c"
		46	xZ	0000	'		
		47	xP	3500	'	☒	"o"
		48	xZ	0000	'		
		49	xP	2100	'		"d"
		50	xZ	0000	'		
		51	xP	3700	'	☒	"e"
		52	xZ	0000	'		
		53	xZ	0000	'		stop
		54	U	0000	'		
		55	xP	0000	'	☒	Input one hex. word
		56	xI	0000	'		for "hex. fill"
		57	[		'	$n_1 n_2 n_3 C m_1 m_2 m_3 m_4$	
		58	B	0257	'		modify address & test
		59	S	0045	'	☒ 1@11-1@29	
		60	T	0000	'		→ Finished
		61	U	0242	'		
		62			'		
		63	xR	0000	'	☒ 3@15 (0241)	

' Conditional Stop Code ☒ Carriage Return

HEXADECIMAL INPUT ROUTINE
(Program 10.1)

FUNCTION:

1. To read hexadecimal information and store it on the memory drum.

2. To verify that the information has been correctly stored by generating a summation of the binary bits stored on the drum (a check sum) and checking this summation against a previously computed summation placed on the input tape.

INPUT:

A tape prepared in its entirety by program 13.1. This tape contains the following:

1. An identification word in the form v n_1 n_2 n_3 m_1 m_2 m_3 m_4 where $N = (n_1 \; n_2 \; n_3)$ = the number of words in hexadecimal to be placed on the drum. $M = (m_1 \; m_2 \; m_3 \; m_4)$ = the initial location in hexadecimal to begin storing the words.

2. N hexadecimal words, each followed by a conditional stop code.

3. The check sum.

PROCEDURE:

1. Transfer to the beginning of this routine with the previously prepared tape in the reader.

2. The routine will read the identification word and set up the initial address and a tally from M and N respectively.

3. The hexadecimal words are read and stored sequentially on the memory drum. After each word is stored the address (for the next word) is incremented by 1 and the tally is decremented by 1.

4. After all hexadecimal words on tape have been placed in memory, the check sum is read in and stored within this routine. Then another check sum is computed in the identical manner used by program 13.1.

5. The computed check sum is subtracted from the one placed on tape. If they are equal, the routine returns for another identification word.

6. If the two check sums are not equal the routine prints "error" and halts. To re-read the same record, back up the tape in the reader to the last "v code" (identified by a punch in channel 6) and depress

(Hexadecimal Input Routine continued)

the start button. The routine will return to re-read the identification word and proceed from there.

OUTPUT:

The information on the tape stored in memory and checked for validity.

TIME:

Reading from tape - one track per minute.
Computing check sum - ten tracks per minute.

STORAGE:

96 locations of instructions and constants.
No temporary storage.

LGP-30 CODING SHEET

Job No. ______ Prog. No. 10.1 Prep. by [illegible] Ck'd. by ______ Date 3-21-57

Problem HEXADECIMAL INPUT ROUTINE Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
;0000	'					
/0000	'	☒				
		0000	xP0050	'		Read I.D. word
		01	xI0000	'		
		02	H0024	'		
		03	E0018	'	☒ 0WWW0000	Drop address
		04	H0019	'		
		05	N0020	'	14@31	
		06	U0009	'		
,00000002	'	07	[ ]	'	☒ NcM	(0021)(0028)
		08	WWWWJ	'	1@11-1@29	(0029)
		09	S0059	'		
		10	A0024	'		
		11	A0019	'	☒	
		12	H0027	'		
		13	U0021	'		
,00000007	'	14	40000000	'	1@1 (0042)	
		15	4	'	☒ 1@29 (0057)	
		16	3WWWWWWQ	'	1@1-1@30 (0044)(0051)	
		17		'		
		18	WWW0000	'	mask (0004)	
		19	[ ]	'	☒ Temp.	
		20	Q	'	14@31 (0005)	
		21	C0007	'		
		22	xP0001	'		
		23	U0026	'	☒	
		24	[ ]	'		
		25		'		
		26	xI0000	'		
		27	[ ]	'	☒ NcM	
		28	B0007	'		
		29	S0008	'	1@11-1@29	
		30	T0102	'		
		31	U0012	'	☒	

' Conditional Stop Code ☒ Carriage Return

LGP-30 CODING SHEET

Job No. ______ Prog. No. 10.1 Prep. by [illegible] Ck'd. by ______ Date 3-21-57

Problem HEXADECIMAL INPUT ROUTINE Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	☒				
		0032		'		
		33	C 0019	'		Store tape ck. sum
		34	C 0027	'	0 → Σ	
		35	B 0007	'	☒	
		36	Y 0127	'	L(Lf)	
		37	U 0038	'		
		38	B 0024	'	L(Lo)	
		39	Y 0040	'	☒	
		40	B[ ]	'		
		41	T 0106	'		
		42	S 0014	'	1@1	
		43	T 0124	'	☒	
		44	S 0016	'	1@1 - 1@30	
		45	T 0124	'		
		46	U 0047	'		
		47	A 0126	'	☒ 1@30	
		48	A 0027	'	Σ	
		49	S 0014	'	1@1	
		50	T 0054	'		
		51	S 0016	'	☒ 1@1 - 1@30	
		52	T 0054	'		
		53	U 0055	'		
		54	A 0119	'	1@1	
		55	C 0027	'	☒ Σ	
		56	B 0040	'		
		57	A 0015	'	1@29	
		58	U 0061	'		
20000002	'	59	7003 0000	'	☒ (0009)	
		60	4000 0000	'	1@1 (0124)	
		61	Y 0040	'		
		62	S 0127	'	B[(Lf) + 1@30]	
		63	T 0040	'	☒	

' Conditional Stop Code ☒ Carriage Return

LGP-30 CODING SHEET

Job No.______ Prog. No. 10.1 Prep. by [illegible] Ck'd. by ______ Date 3-21-57

Problem HEXADECIMAL INPUT ROUTINE Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	☒				
		0100	U0108	'		
		01		'		
		02	XP0024	'		} Read check sum
		03	xI0000	'	☒	from tape
		04	U0033	'		
		05		'		
		06	A0128	'	1@1	
		07	U0043	'	☒	
		08	B0019	'	Check sum	from tape
		09	S0027	'	Computed	check sum
		10	T0114	'		
		11	S0126	'	☒ 1@30	
		12	T0000	'	→ Check	sums are equal
		13	U0114	'		
2,000 0,0009	'	14	8r500	'	Print "e"	Here: ck. sum #1 ≠ ck. sum #2
		15	[ ]	'	☒	
		16	80K00	'	"r"	
		17	[ ]	'		
		18	80K00	'	"r"	
		19	40000000	'	☒ 1@1 (0054)	
		20	8r300	'	"o"	
		21	[ ]	'		
		22	80K00	'	"r"	
		23	U0129	'	☒	
		24	A0060	'	1@1	
		25	U0048	'		
2,000 0,0005	'	26	2	'	1@30 (0047)	(0111)
		27	10002	'	☒ B[(4f)+1@30]	
		28	40000000	'	1@1 (0106)	
		29	[ ]	'	Delay	
		30	[ ]	'	Stop	
		31	U0000	'	☒	

' Conditional Stop Code ☒ Carriage Return

ROYAL MCBEE CORPORATION
ELECTRONIC COMPUTER DEPARTMENT

LGP-30 PROGRAM INPUT ROUTINE
(Program 10.4)

When the overall coding for a problem is surveyed, it is found that the instructions separate logically into independent groups, some of which can be used in any number of problems. Examples of these groups are subroutines of all types, standard input and output routines, and the mathematical subdivisions of the problem. It would be desireable to code these pieces without reference to the other pieces. In order to separate these pieces completely, it is necessary to assign a group of instructions a block of storage locations which does not correspond to actual memory locations; otherwise two blocks of coding might be found to occupy the same section of storage, requiring a change in the coding for one of the two pieces. The "Set Modifier" code word was intended to facilitate this type of coding. A group of instructions can be coded without reference to actual memory locations by starting that group at symbolic address 0000. Then by "setting the modifier" to the "start fill" location, the programmer may position a routine to any part of memory. An instruction preceded by an "x" will not be incremented, and this instruction will still refer to an absolute memory location. It should be noted that instructions may be coded for actual locations merely by setting the modifier to zero (i.e., input code word /0000000). Thus no particular restrictions are imposed upon the programmer by this system.

If the Program Input Routine detects an erroneous code word it will return the typewriter carriage, print "error", and stop. The last word read from tape (or keyboard) contains the erroneous code.

A tape prepared for this Program Input Routine must contain typewriter format control. It is suggested that a carriage return be placed after every four or six words on tape. If there is no format control and the typewriter carriage is permitted to space into the automatic carriage return tab a stop may result. The computer will continue unaffected if the carriage return key is depressed.

TIME:

Instructions are loaded and converted to binary at the rate of one track every 60 - 70 seconds. The Hex. fill code loads and computes the check sum at the rate of one track every 50 - 60 seconds.

STORAGE:

Locations 0000 through 0263. (3 tracks). ~~No temporary storage.~~ 6300

PROGRAM STOPS:

Location	Meaning
0062	Erroneous input code word.
0062	The computed check sum is not identical with the check sum on tape.

PROGRAM INPUT ROUTINE CODE WORDS
(Program 10.4)

Code Order Track Sector	INTERPRETED AS:
$\emptyset\ T_1\ T_2\ S_1\ S_2$	Instruction - modified
$X\ \emptyset\ T_1\ T_2\ S_1\ S_2$	Instruction - Not modified
$8\ 0\ 0\ \emptyset\ T_1\ T_2\ S_1\ S_2$	Negative instruction - modified
$8\ 0\ X\ \emptyset\ T_1\ T_2\ S_1\ S_2$	Negative instruction - Not modified
$+\ 0\ 0\ \emptyset\ T_1\ T_2\ S_1\ S_2$	Command. This word is treated as an instruction to the input routine, using the following word as data. Following word is in hex.
$;\ 0\ 0\ 0\ T_1\ T_2\ S_1\ S_2$	Start fill - The first input word will be stored in location $T_1\ T_2\ S_1\ S_2$, and succeeding words will be stored sequentially
$/\ 0\ 0\ 0\ T_1\ T_2\ S_1\ S_2$	Set modifier - Set "modifier" location to $T_1\ T_2\ S_1\ S_2$. Add modifier to all succeeding instructions not preceded by an "x".
$.\ 0\ 0\ 0\ T_1\ T_2\ S_1\ S_2$	Stop and transfer - Stop (unless BREAK POINT 32 is down) and transfer to $T_1\ T_2\ S_1\ S_2$ upon depression of START button or START COMP. lever.
$,\ 0\ 0\ 0\ 0\ 0\ N_1\ N_2$	Hexadecimal words. The next $N_1\ N_2$ (dec.) words are hex. fill sequentially $1 \leq N_1\ N_2 \leq 63$
$v\ n_1\ n_2\ n_3\ m_1\ m_2\ m_3\ m_4$	Hexadecimal fill - Fill $n_1\ n_2\ n_3$ (hex) words hexadecimally beginning in location $m_1\ m_2\ m_3\ m_4$ (hex.). $1 \leq n_1\ n_2\ n_3 \leq (2047)$. Compute a check sum for these filled locations and verify this computed check sum against a previously computed check sum placed on the tape after the last word on tape. Do not read and verify the check sum if the TRANSFER CONTROL button is depressed.

$\emptyset$ represents any one of the 16 LGP-30 orders (or commands) $T_1\ T_2$ are the decimal digits for the track part of an address. $S_1\ S_2$ are the decimal digits for the sector part of an address

ROYAL MCBEE CORPORATION
ELECTRONIC COMPUTER DEPARTMENT

LGP-30 PROGRAM INPUT ROUTINE
(Program 10.4)

Original Program 10.3 Modified by Rice Institute

The primary function of this routine is the entry of programs from paper tape or typewriter keyboard to storage on the LGP-30 memory drum. The general characteristics of the LGP-30 are described in the LGP-30 Programming Manual. A program consists of two types of words, instruction and data. This routine deals exclusively with the inputing of instruction words and hexadecimal representations of data words. This routine does not handle data words expressed in decimal form.

There are several functions to be performed by a useful program input routine.

1. The most direct way of entering instructions into the LGP-30 is to present it with binary words. But since it is difficult to program in this number system, we prefer to do our programming in decimal notation. If we are to write instructions in decimal form, we must provide the machine with a means of converting such instructions into binary form.

2. Most programs contain instructions which refer to locations within that program. Hence if we wish to place the program in another portion of memory, we must modify some of these instructional addresses.

3. It is sometimes convenient to express a number in binary form. e.g., π or other universal constants.

4. It may be necessary to make instructional or data changes to a program that has already been stored in memory.

These are the functions which this input routine is designed to perform.

This routine recoginzes seven types of code word. After a code word has entered the accumulator (from tape or typewriter keyboard), the accumulator sign position and high order 3 binary bits are analyzed to identify the type of code word. Then a transfer to the appropriate interpretive subroutine within the program input routine is executed. These seven code words and their symbols are as follows.

1. Instruction (none) consists of an order and decimal address. The address consists of two decimal digits for track and two decimal digits for sector. The entire instruction is converted to its binary equivalent and stored in a given location. The address portion is incremented by the contents of the "modifier" (to be discussed below) unless and "x" precedes the order. e.g., b4000 will be incremented: xb6310 will not be incremented. The x will not appear in the stored instruction.

2. Command (+). This code word will be treated as an instruction to the Program Input Routine. The instruction will be executed after the entry of another word. The command code word is input in decimal and is not incremented by the modifier. The second word, presumably data, is assumed to be in hexadecimal. e.g., the command +00h 1637 followed by 73W08 will cause the hexadecimal word 00073W08 to be held (stored) in memory location track 16, sector 37.

3. Start fill (;). This code word tells the input routine where to begin filling input words (instructions and/or hexadecimal representations of data). Each succeeding word will be stored consecutively until or unless a new start fill interrupts the procedure. The address portion of the start fill code

word is decimal, and consists of both a track and a sector. e.g., The code word ; 0003128 will cause the first input instruction or hexadecimal data word to be stored in track 31, sector 28, the second in location 3129, the third in 3130, etc.

4. Set modifier (/). The magnitude of the address of the set modifier code word is stored in a memory cell within the Program Input Routine known as the "modifier". This modifier is added to the address portion of all subsequently input instructions not preceded by an "x". The modifier remains unaltered until replaced by another set modifier code word. The set modifier code word will usually follow a start fill code and will usually be identical to it in magnitude. This word is for use by the Program Input Routine only. It will appear no where else on the drum as such.

5. Stop and transfer (.). This code word is executed in two parts. The first part causes the computer to execute a stop instruction. The second part, initiated by a depression of either the computer START button or the typewriter START COMP. lever, will cause the computer to transfer control to the memory location contained in the stop and transfer code word. The "stop" part of the stop and transfer code is ignored if BREAK POINT 32 on the computer console is in the "down" position. e.g., The code word .0001700 will stop reading. Then, upon depression of the START button, control will be transferred to memory location 1700.

6. Hex. words (,). This code word causes the next $N_1 N_2$ words to be filled without decimal translation. $N_1 N_2$ is specified in the sector portion of the "Hex. words" code word. $1 \le N_1 N_2 \le 63$. e.g., The code word ,0000014 means that the next 14 input words are to be stored in the next 14 locations of memory. The input words must be in hexadecimal and they will not be incremented by the modifier.

7. Hex. fill (v). This code word causes the next n hexadecimal words to fill consecutively beginning in location m. Both m and n are proper hexadecimal numbers. During the filling procedure a summation of the binary bits actually stored into memory (a check sum) is generated. After all (m) words have been stored on the drum, and if the TRANSFER CONTROL button is not depressed, this check sum is compared against a previously computed check sum * placed at the end of the m hexadecimal words. If the two check sums are identical the hex. fill procedure is successfully completed. If the two check sums are not identical the Program Input Routine will return the typewriter carriage and print "error". The format of the Hex. fill code word is $vn_1\ n_2\ n_3\ m_1\ m_2\ m_3\ m_4$ e.g., The code word v1j02w00 will cause the next 1j0=n words $[(1J0)_{16} = (448)_{10}]$ consecutively beginning in location 2W00=m $[(2W00)_{16} = \text{location } (4710)_{10}]$. As many as $(7WW)_{16} = (2047)_{10}$ words can be filled by a single Hex. fill code word.

Leading zeros need not be punched on any input word. All other zeros must be punched. e.g., The code word /0001357 must be completely punched; 000B3749 only the last 5 characters need be punched. i.e., B3749.

* Either program 13.1 or 13.2 will prepare the previously computed check sum and place it at the end of the Hex fill input words.

Job No. ______ Prog. No. 10.4 Prep. by ______ Ck'd. by ______ Date ______

Problem PROGRAM INPUT #3 Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	☒				
		0000	C 0143	'	← 123, 128, 141, 204, 205, 212	
		01	XP 0044	'		
		02	XI 0000	'		
		03	T 0132	'	☒ → NEG. TEST OR LESS	
		04	S 0047	'	1 @ 3	
		05	T 0034	'	→ INST.	
		06	S 0049	'	1 @ 3	
		07	T 0134	'	☒ → ERROR	
		08	S 0144	'	1 @ 3	
		09	T 0162	'	→ COMMAND	
		10	U 0011	'		
		11	S 0047	'	☒ 1 @ 3	
		12	T 0113	'	→ START FILL	
		13	S 0049	'	1 @ 3	
		14	T 0209	'	→ SET MODIFIER	
		15	S 0144	'	☒ 1 @ 3	
		16	T 0153	'	→ STOP + TRANSFER	
		17	U 0018	'		
		18	S 0047	'	1 @ 3	
		19	T 0248	'	☒ → N HEX WORDS	
		20	H 0039	'		
		21	E 0100	'	07WW0000	DROP ADD.
		22	N 0137	'	1 @ 27	
		23	S 0002	'	☒ 1 @ 13 FORM	(n-1) H0000
		24	C 0203	'		
		25	C 0263	'	0 → Σ	
		26	B 0048	'	00003WWJ	
		27	E 0039	'	☒ DROP n	n₂ n₃
		28	U 0031	'		
20000002	'	29	4	'	1 @ 29 + STOP	
		30	27KWKWJ	'		
		31	A 0203	'	☒	

' Conditional Stop Code ☒ Carriage Return

Royal McBee Corporation

Job No. ______ Prog. No. 10.4 Prep. by ______ Ck'd. by ______ Date ______

Problem PROGRAM INPUT ROUTINE #3 Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	☒				
		32	U0054	'		
		33	XZ0000	'	MODIFIER	
		34	R0063	'		
		35	U0050	'	☒	
		36	S0101	'	9 @ 11	
		37	T0102	'		
		38	U0105	'		
		39	[ ]	'	☒ TEMP.	
		40		'		
		41	R0063	'		
		42	U0050	'		
		43	S0101	'	☒ 9 @ 11	
		44	T0103	'		
		45	A0217	'	80000000	
		46	U0105	'		
7 0000 0003	'	47	10000000	'	☒	
		48	3WWJ	'		
		49	10000000	'		
		50	N0029	'	1 @ 29	
		51	E0030	'	☒ TRIM WORD	
		52	H0110	'	N_1	
		53	U0057	'		
		54	Y0226	'		
		55	H0225	'	☒	
		56	U0222	'		
		57	E0129	'	3J3J3J0	
		58	M0130	'	−6 @ 4	
		59	A0110	'	☒ N_1	
		60	H0039	'	N_2	
		61	U0158	'		
7 0000 0004	'	62	2	'	1 @ 30	
		63	U[ ]	'	☒	

' Conditional Stop Code ☒ Carriage Return

Job No.________ Prog. No. 10.4 Prep. by________ Ck'd. by________ Date________

Problem PROGRAM INPUT ROUTINE #3 Track________

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	☒				
		0100	7WW0000	'		
		01	9000000	'	9@11	
		02	A0145	'	9@11	
		03	E0246	'	☒ 800W3WWJ	
		04	A0033	'	MODIFIER	
		05	C[6300]	'	"START FILL" CTR.	
		06	R0149	'		
		07	A0029	'	☒ 1@29	
		08	U0126	'		
70000003	'	09	WWWWWW00	'		
		10	[ ]	'	TEMP.	error in check sum here!
		11	700000000	'	☒	
		12	U0062	'		
		13	R0063	'	HERE FOR START FILL	
		14	U0050	'		
		15	U0126	'	☒	
		16	T0134	'	→ ERROR	
		17	S0260	'	00000002	
		18	U0205	'		
		19	Y0105	'	☒	
		20	Y0149	'		
		21	S0143	'	CTR	
		22	T0124	'		
		23	U0000	'	☒	*
		24	C0203	'	0 → Acc.	
		25	U0147	'		
		26	Y0105	'		
		27	Y0149	'	☒	
		28	U0000	'		*
70000003	'	29	3J3J0	'		
		30	K00000000	'		
		31	F00000000	'	☒	

' Conditional Stop Code ☒ Carriage Return

LGP-30 CODING SHEET

Page 4 of 6

Job No. ______ Prog. No. 10.4 Prep. by ______ Ck'd. by ______ Date ______

Problem PROGRAM INPUT ROUTINE #3 Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	☒				
		1 32	A 0111	'	70000000	
		33	T 0041	'		
		34	XP 1606	'	CARR. RETURN	
		35	XZ 0000	'	☒	
		36	XP 3701	'	"e"	
		37	XZ 0004	'	DELAY + 1@27	
		38	C 0110	'	0 → ACC.	
		39	U 0213	'	☒	
		40	C 0225	'		
		41	800 T 0000	'		*
		42	U 0252	'		
70000003	'	43	[ ]	'	☒ TEMP.	
		44	10000000	'	1@3	
		45	900000	'		
		46	C 0143	'		
		47	XP 0026	'	☒	
		48	XI 0000	'		
		49	XC [6300]	'	"START FILL" CTR.	
		50	B 0029	'	1@29	
		51	A 0105	'	☒	
		52	U 0119	'		
		53	R 0063	'	HERE FOR STOP + TRANSFER	
		54	U 0050	'		
		55	Y 0157	'	☒	
		56	XZ 3200	'		
		57	U [ ]	'		
		58	E 0109	'	0wwwwww00	
		59	M 0131	'	☒ -3/4 @0	
		60	A 0039	'	N_2	
		61	U 0063	'		
		62	R 0063	'	HERE FOR COMMAND	
		63	U 0050	'	☒	

' Conditional Stop Code ☒ Carriage Return

Royal McBee Corporation

LGP-30 CODING SHEET

Job No. ______ Prog. No. 10.4 Prep. by ______ Ck'd. by ______ Date ______

Problem PROGRAM INPUT ROUTINE #3 Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	☒				
		0200	C0203	'		
		01	XP0044	'		
		02	XI0000	'		
		03	[ ]	'	☒ Temp.	
		04	U0000	'		*
		05	T0000	'		*
		06	U0134	'	→ Error	
,00000002	'	07	WWWWJ	'	☒ 1@11 – 1@29	
		08		'		
		09	R0063	'	Here for set modifier	
		10	U0050	'		
		11	~~Y0020~~ Y0033	'	☒	
		12	U0000	'		*
,00000003	'	13	80KQ0	'	"f"	
		14	400000000	'	1@1 + Delay	
		15	80KQ8	'	☒ "f"	
		16	Y0063	'		
,00000004	'	17	800000000	'	1@0	
		18	823F0	'	"0"	
		19	400000000	'	☒ 1@1 + Delay	
		20	80KQ0	'	"f"	
		21	U0111	'		
		22	C0258	'		
		23	XP0002	'	☒	
		24	XI0000	'		
		25	[ ]	'	NHM	
		26	B[ ]	'		
		27	T0256	'	☒	
		28	S0214	'	1@1	
		29	T0261	'		
		30	S0259	'	1@1 – 1@30	
		31	T0261	'	☒	

Conditional Stop Code ' ☒ Carriage Return

LGP-30 CODING SHEET

Job No. ______ Prog. No. 10.4 Prep. by ______ Ck'd. by ______ Date ______ Page 6 of 6

Problem PROGRAM INPUT ROUTINE #3 Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	☒				
		232	U0233	'		
		33	A0062	'	1@30	
		34	A0263	'	Σ	
		35	S0214	'	☒ 1@1	
		36	T0240	'		
		37	S0259	'	1@1 − 1@30	
		38	T0240	'		
		39	U0241	'	☒	
		40	A0219	'	1@1	
		41	C0263	'		
		42	S0207	'	1@11 − 1@29	
		43	A0258	'	☒	
		44	T0140	'	→ FINISHED LOADING	
		45	U0054	'		
0000 0001		46	800W3WWJ	'		
		47	E0240	'	☒	
		48	R0063	'	HERE N HEX WORDS	
		49	U0050	'		
		50	A0105	'		
		51	U0146	'	☒	
		52	XP0031	'	} READ CK. SUM	
		53	XI0000	'	} FROM TAPE	
		54	S0263	'	COMPUTED CK. SUM.	
		55	U0116	'	☒	
		56	A0214	'	1@1	
		57	U0229	'		
0000 0003'		58	[ ]	'	NHM	
		59	3WWWWWWQ	'	☒	
		60	2	'	1@30	
		61	A0219	'	1@1	
		62	U0234	'		
		63	[ ]	'	☒ CHECK SUM	

' Conditional Stop Code ☒ Carriage Return

DATA INPUT #1 SUBROUTINE
(PROGRAM 11.02)

FUNCTION:

To read a decimal number from tape, convert to binary, scale to the proper binal point location, and store the word in a specified drum location. For each number the following is punched on tape:

1. The decimal point location of the number on tape, counting from right to left. (One decimal digit designated as "P").

2. The binal point location desired for the number to be placed on drum. (Sign and two decimal digits designated as "q").

3. The drum location to which the number is to be sent. (2 decimal digits for track and 2 decimal digits for sector).

4. The number to be entered (Seven decimal digits plus sign).

INPUT:

For each word to be stored, an identification word (parts 1, 2, & 3 above), and the signed number (part 4 above) are required.

CALLING SEQUENCES:

Loc.	Inst.	Add.
α	R	$(L_o + 8)_{10}$
$\alpha + 1$	U	L_o
$\alpha + 2$	etc.	

OUTPUT:

The scaled binary representation of the number will appear at its proper drum location.

EXIT:

A "zero" indentification word will cause the routine to exit to (α+ 2).

PROGRAM 11.0R

SCALING:

The location of the decimal point in the decimal number is specified by a number P, which denotes the number of places following the point in the seven digit field. P can be in the range $0 \leq P \leq 9$. The location of the binary point in the full 30 bit binary word is specified by q, the number of digits preceeding the point in the full word. q can lie in the range given in the following table.

In order for the number to be representable at a given q, the number must be less (in absolute value) than 2^q. However, if too large a q is used, the number will not reconvert exactly on output, since there will be too few binary digits following the point to adequately represent the fractional part of the number. The following table also gives the maximum conversion correspondence between P and q.

TABLE OF P vs q:

P	Max q	Min q	Max q for Exact Reconversion	Min q for Max No. Size (all 9's)
0	+47	+2	+30	+24
1	+43	-2	+26	+20
2	+40	-5	+23	+17
3	+37	-8	+20	+14
4	+34	-11	+16	+10
5	+31	-14	+13	+7
6	+28	-17	+10	+4
7	+24	-21	+6	0
8	+21	-24	+3	-3
9	+17	-28	0	-6

PROGRAM 11.0R

TIME:

20 - 25 words per minute.

ACCURACY:

The scaled number in memory may be inaccurate in the 30th binary position.

STORAGE:

192 locations of instructions and constants. Eight locations of temporary storage (Track 63, sectors 03, 04, 45, 52, 54, 55, 56, 57).

PROGRAM STOPS:

Loc.	Meaning
$(L_0 + 0234)_{10}$	Divide check in scaling data word. $\lvert N \rvert \geq 2^q$.

EXAMPLES: (See LGP-30 Data Input 1 load sheet)

1. Place +96.40236 in drum location 6234 at a q of +7
2. Place -.000000597 in drum location 2363 at a q of -14
3. Place +330000. in drum location 2100 at a q of +30

TAPE PUNCHING INSTRUCTIONS:

1. All characters of the I.D. word must be punched. e.g., punch 0+096300' completely. The first three characters should not be omitted. The stop code (') must be the last character.
2. Leading zeros of a positive number need not be punched. To enter all zeros merely punch a stop code. All digits of a negative number must be punched.
3. Be sure to check each load sheet to see whether an additional stop code should follow the last number punched.

LGP-30 DATA LOAD SHEET

Job No.______ Prog. No. 11.0R Prep. by______ Ck'd. by______ Date______

Problem EXAMPLES FOR DATA INPUT #1 Case______

INPUT #1

Quan.	P	± q	Location	Stop	± Number	Stop	Car. Ret.
	5	+07	6234		9640236		
	9	-14	2363		-0000597		
	0	+30	2100		330000		

$0 \le P \le 9$

$-28 \le q \le 47$

$0000 \le Loc \le 6363$

Punch a stop code after the last number? Yes X No______

Job No. ______ Program No. 11.0R Prepared by M. Kaye Date 10/24/56

Problem DATA INPUT #1 SUBROUTINE Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
;000	'					
1000	'	☒				
		0000	B 0115	'	A0030	
		01	C 0044	'		
		02	x P 0038	'		Input I.D. word
		03	x I 0000	'	☒	
		04	x H 6354	'	Orig. I.D. word	
		05	T 0022	'		→ P > 7
		06	U 0007	'		
		07	S 0029	'	☒ 1@30	
		08	T [ ]	'	Exit	Out if I.D. word = 0
		09	A 0145	'	1@30	
		10	N 0132	'	1@29	Shift left 2
		11	E 0233	'	☒ 05WKWKWJ	Drop P + trim I.D. word
		12	x H 6355	'	N_1	
		13	U 0017	'		
		14	M 0136	'	1@24	shift rt. 24 bits
		15	U 0044	'	☒	
		16	x Z 0030	'	30@29 (0051)	
		17	E 0060	'	03J3J3J0	
		18	M 0140	'	-6@4	
		19	x A 6355	'	☒ N_1	
		20	x H 6356	'	N_2	
		21	U 0025	'		
		22	R 0044	'		
		23	U 0010	'	☒	
		24	x Z 0032	'	8@27 (0044)	
		25	E 0240	'	0WWWWW00	
		26	M 0262	'	-3/4@0	
		27	x A 6356	'	☒ N_2	
		28	U 0031	'		
;0000 0002	'	29	2	'	1@30 (0007)(0100)	
		30	[ ]	'	(0044)	
		31	Y 0260	'	☒ loc. of clear of No. to memory	

' Conditional Stop Code ☒ Carriage Return

Rev. 3-20-57

LGP-30 CODING SHEET

Job No. ______ Program No. 11.0R Prepared by M. Kaye Date 10/24/56

Problem DATA INPUT #1 SUBROUTINE Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	☒				
		0032	S 0247	'	1@7	
		33	T 0220	'		→ positive q
		34	E 0213	'	wwww0000	here for neg. q. Drop add.
		35	xC 6357	'	☒	Complement q
		36	xS 6357	'		
		37	M 0059	'	1@14	Shift to 29
		38	xH 6305	'	q @29	
		39	xB 6354	'	☒ Orig. I.D. word	
		40	E 0219	'	70000000	Drop all but P@3
		41	U 0014	'		
		42	B 0121	'	exit #0114	here for positive no.
		43	U 0108	'	☒	
		44	A [ ]	'	A0030 or A0024	
		45	U 0046	'		
		46	xH 6304	'	4P@29	
		47	M 0133	'	☒ 1@2	P@29
		48	A 0113	'	$L_0(10^P)$	
		49	Y 0235	'	$L(10^P)$	
		50	Y 0222	'	$L(10^P)$	
		51	B 0016	'	☒ 30@29	
		52	xS 6303	'	q@29	
		53	xS 6304	'	4P@29	
		54	xP 0012	'		Start reader
		55	T 0227	'	☒	→ shift right to scale
		56	R 0114	'		here: shift left to scale
		57	U 0061	'		
		58	U 0234	'		
20000002	'	59	20000	'	☒ 1@14 (0037)	
		60	3J3J3J0	'	(0017)	
		61	A 0226	'	$D(L_0)$	
		62	C 0234	'	scale order	
		63	xI 0000	'	☒	Input data word

' Conditional Stop Code ☒ Carriage Return

Rev. 3-20-57

Job No. ______ Program No. 11.0R Prepared by M. Kaye Date 10/24/56

Problem DATA INPUT #1 SUBROUTINE Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	⊠				
		0100	N 0029	'	1@30	Shift out of spacer
		01	E 0237	'		Trim word
		02	x H 6352	'	No. in dec.	
		03	S 0239	'	⊠ 1@30	
		04	T 0259	'		→ word = 0
		05	S 0141	'	1@2 - 1@30	
		06	T 0042	'		→ positive word
		07	B 0243	'	⊠ Z0216	here for neg. word
		08	Y 0144	'	binarization exit	
		09	x B 6352	'	No. in dec.	
		10	E 0246	'	1wwwwwwQ	drop ± position
		11	x H 6354	'	⊠ N_1	
		12	U 0116	'		
		13	Z 0248	'	L_0 (10^P) (0048)	
		14	U [ ]	'	U0058 or U0230	
		15	A 0030	'	⊠ (0000)	
		16	E 0238	'	01Q1Q1Q0	
		17	M 0146	'	-6@4	
		18	x A 6354	'	N_1	
		19	x H 6355	'	⊠ N_2	
		20	U 0124	'		
		21	Z 0114	'	(0042)	
		22		'		
		23	M 0148	'	⊠	
		24	E 0139	'	01Q0WQ00	
		25	M 0147	'	-156@8	
		26	x A 6355	'	N_2	
		27	x H 6356	'	⊠ N_3	
		28	E 0143	'	1wwQ0000	
		29	M 0215	'	-55536@16	
		30	U 0134	'		
9000 0003	'	31		'	⊠	

' Conditional Stop Code ⊠ Carriage Return

Rev. 3-20-57

LGP-30 CODING SHEET

Page 4 of 6

Job No. ______ Program No. 11.0R Prepared by [illegible] Date 10/24/56

Problem Data Input #1 Subroutine Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	☒				
		0132	4	'	1@29 (0010)	
		33	20000000	'	1@2 (0047)	
		34	XA6356	'	N_3	
		35	U0144	'	☒	
		36	XZ0032	'	1@24 (0014)	
		37	B0123	'	M(Lo)	
		38	U0241	'		
20000041	'	39	1Q01WQ00	'	☒ (0124)	
		40	K0000000	'	(0018)	
		41	1WWWWWWQ	'	1@2 - 1@30 (0105)	
		42		'		
		43	1WWQ0000	'	☒ (0128)	
		44	U[ ]	'	U0114 or U0216	
		45	2	'	1@30 (0009)	
		46	K0000000	'	-6@4 (0117)	
		47	G2000000	'	☒ -156@8 (0125)	
		48	7WWWWWWQ	'	.9999999999@0	Start of table of shifts
		49	40000000	'	1@1	
		50	20000000	'	1@2	
		51	10000000	'	☒ 1@3	
		52	8000000	'	1@4	
		53	4000000	'	1@5	
		54	2000000	'	1@6	
		55	1000000	'	☒ 1@7	
		56	800000	'	1@8	
		57	400000	'	1@9	
		58	200000	'	1@10	
		59	100000	'	☒ 1@11	
		60	80000	'	1@12	
		61	40000	'	1@13	
		62	20000	'	1@14	
		63	10000	'	☒ 1@15	

' Conditional Stop Code ☒ Carriage Return

Rev 3-20-57

Job No. ______ Program No. 11.0R Prepared by M. Kaye Date 10/24/56

Problem DATA INPUT #1 SUBROUTINE Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	☒				
		0200	8000	'	1@16	
		01	4000	'	1@17	
		02	2000	'	1@18	
		03	1000	'	☒ 1@19	
		04	800	'	1@20	
		05	400	'	1@21	
		06	200	'	1@22	
		07	100	'	☒ 1@23	
		08	80	'	1@24	
		09	40	'	1@25	
		10	20	'	1@26	
		11	10	'	☒ 1@27	
		12	8	'	1@28	
		13	WWWW 0000	'	(0034)	
		14		'		
		15	9388 0000	'	☒ -55536@16 (0129)	
		16	X C 6352	'		Complement No.
		17	X S 6352	'		
		18	U 0114	'		
3000 0001	'	19	7000 0000	'	☒ (0040)	
		20	A 0263	'	1@7	
		21	U 0037	'		
		22	D [ ]	'	L (10^P)	Divide by 10^r @ 4P
		23	M [ ]	'	☒ scale	Shift right
		24	U 0260	'		
		25		'		
		26	D 0148	'	D(L0) (0061)	
		27	X C 6356	'	☒	
		28	R 0114	'		
		29	U 0137	'		
		30	U 0222	'		
3000 0005	'	31		'	☒	

' Conditional Stop Code ☒ Carriage Return

(Data Input No. 2 Subroutine continued)

Output:

Binary integer at q = 30 in accumulator.

Accuracy:

Conversion is exact.

PROGRAM STOP:

Loc.	Meaning
$(L_0 + 0117)_{10}$	Divide check in scaling data word. $\lvert N \rvert \geq 2^q$

STORAGE:

89 locations of instructions and constants.
No temporary storage.

NOTE:

Leading zeros of a positive number need not be punched. To enter all zeros merely punch a stop code. All digits of a negative number must be punched.

LGP-30 CODING SHEET

Job No. ______ Prog. No. 13.1 Prep. by [illegible] Ck'd. by ______ Date 3-20-57

Problem HEXADECIMAL PUNCH OR PRINT #2 Track ______

Program Input Codes	Stop	Location	Instruction Op. Address	Stop	Contents of Address	Notes
	'					
	'	⊠				
9000 0003	'	0300	4000 0000	'	1@1 (0242) (0249)	
		01		'		
		02	3WWWWWWQ	'	1@1 - 1@30 (0244)(0251)	
		03	A0225	'	⊠ 1@1	
		04	U0248	'		
		05		'		
		06	A0156	'	1@1	
		07	U0243	'	⊠	
		08		'		
		09	U0000	'	(0237)	
		10	U0234	'	(0024)	
9000 0001	'	11	2	'	⊠ 1@30(0246)	
		12		'		
		13		'		
		14		'		
		15		'	⊠	
		16		'		
		17		'		
		18		'		
		19		'	⊠	
		20		'		
		21		'		
		22		'		
		23		'	⊠	
		24		'		
		25		'		
		26		'		
		27		'	⊠	
		28		'		
		29		'		
		30		'		
		31		'	⊠	

' Conditional Stop Code ⊠ Carriage Return