RCA 601 ELECTRONIC DATA PROCESSING SYSTEM

AUTOMATIC ASSEMBLY SYSTEM

A PROGRAMMER'S REFERENCE MANUAL

RADIO CORPORATION OF AMERICA

Electronic Data Processing Division
Camden, New Jersey

RCA 601 ELECTRONIC DATA PROCESSING SYSTEM

AUTOMATIC ASSEMBLY SYSTEM

In the preparation of this manual, a thorough knowledge of the RCA 601 has been assumed on the part of the reader. As this manual is preliminary, it may be expected that additional information regarding the system will be made available in subsequent publications.

RADIO CORPORATION OF AMERICA | Electronic Data Processing Division Camden, New Jersey

P601-03-022

January, 1961

TABLE OF CONTENTS

I. INTRODUCTION

The 601 Assembly System has been designed to serve as the immediate basis for programming the 601 computer. The assembler has been specified to operate within a minimum 601 configuration which includes one memory storage unit (8192 words), six tape stations (66KC), a paper tape or card reader, and an on-line printer. Since the assembler is a machine-oriented programming system, all features of the 601 are reflected.

In addition, the 601 Assembly System is being used as the basis for the development of the total complement of programs to be offered with the 601 computer. This includes a COBOL Narrator, ALGOL, and various utility and program testing routines.

Thus, the 601 Assembly System is a foundation upon which RCA is building a complete line of programming services and upon which the 601 user may depend, not only as an interim measure before delivery of more complex programming aids, but also as an effective means of preparing the solution for all his computer problems.

II. USING THE PROGRAM SHEET

The Assembly System program sheet is used to specify the program instructions and the various Assembly System directives described in Chapter VIII, Descriptor Verbs.

A. DESCRIPTION AND GENERAL USAGE

1. Sequence

The number (1-4 digits) entered in the column marked SEQUENCE is used to control the internal ordering of the pseudocode instructions. Use of this option allows the programmer to develop his coding without concern for exact ordering as the Assembly System will put it into proper order before any actual processing begins.

A sequence number must be entered with each instruction which has been assigned a specific NAME by the programmer. This number is used to develop a more complete control number within the Assembly System.

2. Name

The symbolic name entered in this column is used to indicate a logical group of instructions and to provide a means of internal relative addressing. The entry in the NAME column may consist of up to ten characters, and should provide a mnemonic indication of the function of the instructions being named. Note, however, that certain letters, words, and symbols have been reserved for Assembly System control. These will be described later in this section.

As indicated above, not all instructions need to be named. Any unnamed instruction will receive an internal name which is relative to the last one assigned by the programmer. Thus the instruction following one called SORT would be named internally SORT + 1. It should be noted that, since sequence numbers are assigned only to those instructions carrying a NAME entry, the relative names are assigned in the order of the input, with appropriate internal control numbers being developed from the last assigned sequence number. The unnamed instructions, then, are handled as a group with the last previous named one.

Omitted instructions may be entered into any of these groups by a special usage of the NAME column. If, for example, one had the following sequence of instructions:

SORT	OP_0
	OP_1
	OP_2
	OP_3

and wished to insert a new instruction OP_m between OP_1 and OP_2, that instruction may be named SORT + 1.1 in the NAME column and given the same sequence number as SORT. The resultant ordered sequence will be:

SORT	OP_0
	OP_1
SORT + 1.1	OP_m
	OP_2
	OP_3

Note that internal references to SORT + 2 and SORT + 3 will still go to the lines marked OP_2 and OP_3 respectively. Thus, the relativity of the original instructions has not been disturbed by the insertion of the new one. The value of the digits preceding the point may range from 0 to 9999, whereas the value

NAME ______________________

PROGRAM SHEET
601 ASSEMBLY SYSTEM

PAGE ______
OF ______

SEQ.	NAME	OPERATION	T	N I	COUNT	ADDRESS or CONDITION

following the point may range from 1 to 99. This implies that up to ninety-nine instructions may be inserted between any two existing ones.

3. Operation

This column contains a symbolic representation of the operation codes available with the RCA 601. It is also used for the various Assembly System Descriptor Verbs.

4. The "T" Column

Where appropriate, this column is used to indicate what action is to be taken with the tag-bits of the operands addressed by the instruction.

For all but the input-output instructions, the entry in this column may be either E, U, or D

where:

E = sense origin for tag equal to contents of Tag Register.

U = sense origin for tag unequal to contents of Tag Register.

D = sense origin and destination for tag unequal to Tag Register.

For input-output instructions, the entry in this column may be either E, U, or S

where:

E = sense origin for tag equal to tag specified in P.D. half-word.

U = sense origin for tag unequal to tag specified in P.D. half-word.

S = set tags in destination to tag specified in P.D. half-word.

The absence of an entry where it is required by an instruction, will cause the generation of an "ignore-tag" function in the resultant operation half-word.

5. The "NI" Column

An X entered in this column causes the instruction to be non-interruptable. Leaving this column blank allows an interupt to occur upon the completion of the execution of the instruction.

6. Count

A one to three digit decimal number is entered in this column for those instructions that are controlled by a specified count. The range of the count varies with the particular instruction. (See Order Code Charts.)

7. Address or Condition

This column is utilized for three types of information. First it is used to specify those special conditions normally associated with selected operation codes. For instance, the Store Register instruction requires a register designation to appear in the operation half-word. In using the Assembly System program sheet, a symbolic register designation is entered into the first position of the ADDRESS or CONDITION column. Similarly, the column may be used to express various conditions to be set or sensed. Additional uses for this column are shown in the chapter devoted to the Assembly System order code.

The second, and perhaps most important, function of this column is to contain the symbolic addresses needed by each instruction. The prescribed format for this column requires that an asterisk(*) be placed immediately to the left of each address. This implies that a conditional or register statement, as described above, is not preceded by an asterisk, as is the first actual address. Each instruction requires that at least a standard number of asterisks, corresponding to the number of addresses required by the instruction, appear with the instruction. The requirements for each instruction are given in this manual in the chapter devoted to the order code. If any of the required addresses is to be assumed, the asterisk for that address must appear; however, no symbolic appears with that asterisk. For example, if an instruction requires three addresses, and the B address is to be assumed, the addresses are written as:

*NAME1 ** NAME2

The generated machine address of NAME2 will occupy the second half-word following the operation half-word, with the A digit of the operation half-word being properly adjusted by the Assembly System.

Should one wish to supply more than the required number of addresses for any instruction, except the input-output instructions, and those marked with a "1" in the note column (N) of the order code charts, he may do so by writing three addresses, each preceded by an asterisk. The Assembly System will consider the supplied addresses to be, from left to right, the A, B, and C addresses respectively. Asterisks must appear for each assumed address. The Assembly System will adjust the A digit of the operation half-word accordingly and, unless otherwise specified by the programmer, supply the LHE of the data addressed by the symbolics appearing in the unused addresses.

All input-output instructions must either be written with the standard number of asterisks for the instruction or with four asterisks. If written with four addresses, these addresses will be considered by the Assembly System to be, from left to right, the A, B, and C addresses and the Peripheral Device Half-Word, respectively. Any of the first three addresses may be assumed but their asterisks must be present. The Assembly System will adjust the A digit appropriately and, unless otherwise specified, supply the LHE as explained above.

If the required entries do not conveniently fit on one line, the following lines may be used for the overflow. However, no address may be split over two lines and not more than 100 characters may be entered in the Address or Condition column of any one line.

The third use of the ADDRESS or CONDITION column is for comments. Following the last entry for any instruction, the user may enter a colon(:) and as many as 80 characters of remarks or comments. The comments may be split across succeeding lines.

B. ADDRESS MODIFICATION

Address modification is expressed by following an address with a comma and Mn, where n is a two digit octal number. Address Modifiers 10, 20, 30, 40, 50, 60, and 70 perform the normal function of address modification. However, by writing the second digit as a "1", the programmer may specify that the Address Modifier is a self-incrementing type, i.e., it is to be automatically incremented after each address modification. Thus, Address Modifiers 11, 21, 31, 41, 51, 61, and 71 are the self-incrementing counterparts of those numbered 10, 20, 30, 40, 50, 60, and 70. M01 is reserved for the Lower Limit Register.

An example of address modification would be:

*TAX, M20

where Address Modifier 20 will be applied to the address of TAX before the instruction is executed.

C. INDIRECT ADDRESSING

Indirect addressing may be indicated by a comma, followed by an I. If special control conditions are to appear in the rightmost three bits of the indirect address half-word, they are indicated by a number following the I. This number is composed according to the standard rules found in the RCA 601 General Information Manual.

For example,

*TAX,I5

would indicate that 1) TAX is an indirect address, 2) no further indirect addressing is permitted, and 3) the AM specified in the indirectly addressed location is not to be applied.

Any combination of the above-mentioned symbolics may be used, e.g., *NAME,I; *NAME,M20; *NAME, M20,I5. However, if M is present, it must precede any indirect address indicator.

D. RESERVED CHARACTERS

In the RCA 601 Assembly System, the following characters are not available for use as the first character of any NAME:

&, #, "(quotes), (zero), %, $, /, -

In addition, certain characters may not appear *anywhere* within a NAME,

They are:

*, comma, +, parentheses, space, <, >, ; (semicolon), : (colon)

E. RESERVED WORDS

The following words may not appear in the NAME column of the coding sheet:

INSERT
DELETE
REPLACE
CON-FIX
CON-SEGxx

where the "xx" of CON-SEGxx ranges from 1 to 99.

F. SPECIAL FORMATS

Several special formats are necessary to properly represent each of the functions available with the RCA 601 Assembly System. They are as follows:

1. Machine Code Addresses

A machine code address may be specified in the ADDRESS or CONDITION column by using a number sign (#) followed by eight octal digits as follows:

*#TIHHHHHC

Where:

T = the tag digit (0-7) (See below)

IHHHHH = A six digit octal half-word address (I = 0 or 1)

C = Character address. This digit may take on the values:

0, 1, 2, 3, 4, 5, 6, 7 for three-bit characters

0, 1, 2, 3, 4, 5 for four-bit characters

0, 2, 4, 6 for six-bit characters

0, 2, 4 for eight-bit characters

In addition the machine address can be followed by a ,Mn and/or a ,In notation.

For example,

```
*#30123453
*#60123453,M40
*#60123450,M40,I5
*#60123450,I5
```

The Assembly System will automatically assign tags to half-words according to the following scheme:

1 - an operation half-word

2 - a non-floatable address half-word

3 - Constant half-word

5 - JUMP, JUMP.S, JUMP.R half-word

6 - a floatable address half-word

7 - The address associated with a Set Limit Register Instruction.

This scheme permits editing service routines to distinguish between these different types of half-word in printing programs and, in addition, permits the loading routine to distinguish between floatable and non-floatable half-words. In the assignment of tags to machine code addresses, if the programmer follows this scheme, he can ensure that the edit will print machine code addresses in the proper format. If, however, the programmer wishes to place special tags on certain machine code addresses, he may use tags 0 and 4. These tags will never be automatically supplied by the Assembly System unless specifically requested in a machine address. Upon loading, tag 0 will be considered to indicate a non-floatable half-word; tag 4 will indicate a floatable half-word.

2. Half-Word Instructions

Certain instructions in the RCA 601 occupy only one half-word in machine code. However, these instructions are expressed on the Assembly System program sheet in the normal manner, i.e., with the symbolic operation code in the OPERATION column and the appropriate entries in the ADDRESS or CONDITION column.

3. Absolute Address

Any of the addresses in the full-word Arithmetic instructions may be indicated as addressing absolute quantities by enclosing the entire symbolic address within parentheses.

For example:

```
* (TAX + 14, M20)
```

4. Zero Addresses

A pseudocode address of a single zero (0) generates a 19 bit machine code address of all zeros. This may be used as an address to be modified during the execution of the program. Address Modifiers and indirect address indicators may be used in conjunction with the zero symbolic, where appropriate.

For example:

* 0, M30

5. Accumulator Addressing

For those instructions which may address the accumulator, the ampersand (&) is used as the accumulator address. Enclosing the ampersand in parentheses indicates that the accumulator is addressed and its contents are to be treated as an absolute value.

Examples:

* &

* (&)

Address Modification and Indirect Addressing are not available when addressing the Accumulator.

6. Standard Locations

Special symbolics are provided for all standard locations in the RCA 601. Each of these symbolics is preceded by a dollar sign.

For example, the symbolic address

* $M30

is used when the HSM address of this Address Modifier is to be generated for loading, alteration, etc. A complete list of standard locations may be found in Appendix I.

7. Special Half-Word

Two options are available with the special half-word.

a. *Option 1*

The Special Half-Word may be used to cause the generation of a machine half-word which may then be addressed by the Set Data Register Instruction to set the Control Register or its parts. The format of the machine half-word is as follows:

* /Rxx, Cn, Tnn, Sx

where:

/ - *is the introductory symbol of a Special Half-Word.*

Rxx - *refers to the round indicators. The "R" is the signal for the round indicator. The xx may be either H, W, HW or blank. If H, the generated half-word may be used to set the Half-Word Round Indicator and reset the Word Round Indicator. If W, the generated half-word may be used to set the Word Round Indicator and reset Half-Word Round Indicator. If HW, the generated half-word may be used to set both Round Indicators, and if blank, both Round Indicators may be reset.*

Cn - *refers to the Character Length Register where n may equal 3, 4, 6 or 8, representing the bit size to which the register is to be set.*

Tnn - *refers to the tag register, where nn may be one or two digits. If one digit, it is a number from 0 through 7, representing the tag to be placed in the register. If two digits, the first is as described above and the second is a one, indicating that the tagtransfer option is desired.*

Sx - *refers to the symbol Register, where x is the symbol to be placed into the register. It may be any punchable 4 or 6 bit character with the exception of space, *, >, <, ;, quotes, colon and comma, or may be three octal digits surrounded by quotes. The second option must be used for all 3 or 8 bit characters and for the exceptions noted above.*

Option 1 of the Special Half-Word may be written with any one or any combination of the four elements. The Sx option alone, may be used as the symbol half-word of the Fill instruction.

Examples:

```
* / RH, C4, T31, SA
* / RHW
* / C4
* / T3
* / S "075"
* / R, C8
```

Note that all bit positions of the generated special half-word for which no symbolic was designated will be zero filled. For those symbolics that are designated, the requested bits or characters will be appropriately positioned within the generated half-word. The symbolics of option 1 of the Special Half-Word need not be written in a prescribed order.

b. Option 2.

The Special Half-Word may also create a machine half-word that may be used with the Set Address Register instruction when the Limit Register Condition is specified. In this case the Special Half-Word is written as follows:

```
* / LR LLR, ULR
```

where:

/ - *is the introductory symbol of a Special Half-Word*

LR - *specifies that the Limit Register option of the Special Half Word is desired*

LLR - *specifies the Lower Limit Register setting desired. It may be LLR, in which case the lower limit register setting of the program being assembled is automatically supplied by the assembly system, or it may be specified as a number sign (#) followed by 4 octal digits.*

ULR - *specifies the Upper Limit Register setting desired. It may be ULR, in which case the upper limit register setting of the program being assembled is automatically supplied by the assembly system, or it may be specified by a number sign (#) followed by 4 octal digits.*

If the number sign followed by four octal digits is used, the four octal digits represent the first 4 digits of the address to which the limit is to be set, where the first digit may be one or zero and the last digit must be zero or four. The least significant three digits of the address (not specified with this option) are assumed to be zeros.

Examples:

```
* / LR LLR, ULR
* / LR LLR, #1750
* / LR #0454, #1750
* / LR #0454, ULR
```

The symbolics of option 2 of the Special Half-Word must be written in the order shown above.

8. Peripheral Device Half-Word

All input-output instructions require a Peripheral Device Half-Word. The PD half-word is written in the following format:

```
* PDnn, M, Tn, G, Cn, Sx, F
```

where:

PDnn - *nn specifies the relative peripheral device number (PDnn must always be present).*

M - *if present specifies that the instruction is to be operated in the tag mode. If not present, the instruction is to be operated in the non-tag mode.*

Tn - *if present, n specifies the tag to be sensed or set as specified in the operation half-word.*

G - *if present, indicates that in case of error, the operation is to go to completion before preforming the error jump. If absent, the error jump will be performed immediately upon detection.*

Cn - *The n represents the character size of the information being read or written and may be 3, 4, 6, or 8. A Cn notation may be used to indicate a code conversion where applicable, and* <u>*must*</u> *be present if the Sx is present.*

Sx - *The x represents the symbol that is to control this instruction. It may be any punchable 4 or 6 bit character with the exception of space, *, >, <, ;, quotes, colon and comma; or it may be of the form "nnn" where nnn are three octal digits representing a 3 or 8 bit character, or one of the 4 or 6 bit exceptions noted above. The Sx notation must be present for all symbol controlled input-output instructions.*

F - *if present, indicates monitor printer format control.*

The Peripheral Device half-word can thus consist of seven elements; however, the PDnn element must appear in the first position in every Peripheral Device half-word. The relative sequence of the remaining elements is immaterial. When various elements do not appear, the comma and that element are not written.

Examples:

```
* PD 21
* PD21, M
* PD21, T4, G
* PD21, M, G, C3, S"003"
* PD21, M, T4, G, C6, SA
```

9. Mask Half-Word

The Mask Half-Word may be used to specify a mask for the following instructions:

SENSE.AND

SENSE.OR

SET .DR (MIR,PIR,PIB)

The first character of the Mask half-word is a minus (-). Following the minus, the programmer lists the bits he wishes to have set to "one" by their symbolics, separated by commas. All unlisted bits will be set to zero. The symbolics that may be used are as follows:

VF1	- overflow indicator bit
VF2	- subsidiary overflow indicator bit
UF1	- underflow indicator bit
UF2	- subsidiary underflow indicator bit
ZDI	- Zero Divisor indicator bit
TE	- Tag equal in origin indicator bit
TU	- Tag unequal in origin indicator bit
TD	- Tag unequal in origin or destination indicator bit
LLA	- Lower Limit Alarm indicator bit
ULA	- Upper Limit Alarm indicator bit
PRN	- PRN bit
PRZ	- PRZ bit
PRP	- PRP bit
NFI	- Not found indicator bit
TCC	- Transfer Control Condition indicator bit
IWI	- Instruction Waiting indicator bit
PI1	- Program Indicator 1 bit
PI2	- Program Indicator 2 bit
PI3	- Program Indicator 3 bit
ETI	- Elapsed Time Indicator bit
EI	- External Indicator bit
SIT	- Simultaneous Instruction Termination bit
DO	- Do Indicator bit

Not more than eleven symoblics can be listed in any one Mask half-word. The listed symbolics need not be in any specific sequence.

Example:

```
* ⊖ EI, TCC, PI1, PI2, PRP, PRZ
```

10. Guarded Address

The C address of input-output instructions may specify a guarded jump or may indirectly address a half-word specifying a guarded jump. In the Assembly System one may specify a guarded jump address by enclosing the entire address with quotes. The C digit of such an address will be forced to 1.

Example:

```
* "NAME + 12, M31"
```

III. USING THE DATA SHEET

The 601 Assembly System Data Sheet is used to describe:

File (Read-In) Areas,
Working Storage Areas, and
Constants.

Memory is allocated in the order in which the descriptions appear on the Data Sheets. The user may then refer to file areas and their parts, working storage areas and their parts, and constants by the name which he assigned to them. The Assembly System replaces the names with their appropriate leftmost or rightmost addresses according to the requirements of the instructions in which they are used. Provision is also made in the addressing scheme for overriding the assignment of the leftmost or rightmost address. (See addressing).

A. DESCRIBING FILES

1. Sequence

In the first column labeled "SEQ." a sequence number is entered to provide convenient tags to each line for correction purposes. Each line with an entry in the "DATA NAME" column on the Data Sheet must have an entry in the Sequence column. This entry may range from 0 to 99999. Entries must be numbered consecutively.

2. Data Name

The names assigned to each piece of data are entered under "DATA NAME". They may be alphanumeric but may not exceed ten characters in length.

3. Function

In file and working storage descriptions, the following entries may be made in the "FUNCT" column:

ALOC	Allocate
REN	Rename
DUP	Duplicate

or the column may be blank.

4. Size

The "SIZE" column is used to designate the size of the information element described on the line. It is broken down into three sub-headings, Number, Units, and Repeat.

Under "NO.", a number (0-9999) is entered representing the number of words, half-words, or characters of memory required to store the information element described on the line. Under "U", the units in which the size is expressed is specified by:

W for words,

H for half-words,

C for characters.

Under "REP." a number (0-9999) is entered if this information element appears more than once in the area being defined. The value entered is the number of times this information element appears. This column is convenient to use when a series of items of similar format follow each other in a file. One

DATA SHEET

601 ASSEMBLY SYSTEM

NAME ____________________

PAGE ______

OF ______

SEQ.	DATA NAME	FUNCT.	SIZE			BIT CODE	LEFT ADDR.		DESCRIPTION
			NO.	U	REP.		HALF W	A	
1	A	ALOC		W					
2	CRITERION		2	W					
3	NAME		20	C		6			
4	ADDRESS		40	C		6			
5	AREA CODE		1	H		3			CHARACTER ACCESS REQUIRED
6	DEPOSITS		1	H	10				
7	BC	ALOC					000010	A	
8	DATE	REN	6	C		6			
9	MONTH		2	C		6			
10	DAY		2	C		6			
11	YEAR		2	C		6			
12	SHOP-ORDER		1	H					
13	T	DUP		H	A				

need not make repetative entries for each item, but may describe the first item, and then request that the description be repeated. A procedure is available for addressing each of the repeated items individually.

5. Bit Code

The character size of the element being described is given in the "BIT CODE" column. In file and working storage descriptions, 3, 4, 6, or 8 may be entered.

6. Left Address

The Left Address columns are used to indicate that the element described is to overlay another area or part of another area. Under "HALF-W" a number (6 decimal digits) is entered which represents the half-word address within the overlaid area at which the overlay is to begin. Addressing within each area begins with half-word 000000. The overlaying area is assigned memory starting with the half-word specified by the Left Address entry. The "A" column is used to name the overlaid area. Left Address entries may only be made on ALOC or DUP lines of file and working storages, and the area named in an "A" column of a "LEFT ADDRESS" entry may only be one which has been previously defined and named on an ALOC or DUP line.

7. Description

For file and working storage descriptions, the "DESCRIPTION" column may be used for programmer remarks.

8. Allocating A File Area

The first step in describing a file is to allocate an area of memory. A sequence number is entered under "SEQ.". The "DATA NAME" of the allocation line may be left blank or one or two characters may be entered as the name of the file area. If a file is to be overlaid (through a "LEFT ADDR" entry elsewhere) or duplicated (through a DUP function elsewhere) the "DATA NAME" column may not be left blank. ALOC is entered under "FUNCT".

In the "SIZE" columns, the "NO." and the "REP." entries must be left blank on ALOC lines. The "U" column may be left blank or a W or H may be entered there. Blank or H specifies that allocation of this area is to commence in the next available half-word location without regard for word orientation. W specifies that allocation of this area is to commence in the next available word location. ALOC lines may not have a C entry in the "U" column; however, a "BIT CODE" entry of 3, 4, 6 or 8 must be made if the area is subject to character relative addressing.

If the file being described is not to overlay another allocated area, the "LEFT ADDRESS" columns are left blank.

See sequence no. 1 on the sample Data Sheet for an example of the definition of file areas.

The file area is then further defined by following the ALOC entry with a series of subdivision names. Each of these describes successive areas within the file area. Note that these lines do not require an entry in the "FUNCT." column. The first entry following an ALOC must be defined in units that are less than or equal to the units specified in the ALOC line; i.e., if the "U" column of the ALOC line is left blank or contains an H, the next entry may be defined in terms of H or C; if the ALOC line is defined in terms of W, the next entry may be defined in terms of W, H, or C.

Each of these entries must carry a size indication in terms of the number of words, half-words or characters being described. All entries described in terms of characters must carry a "BIT CODE" entry (3, 4, 6, or 8) as must every area that will be addressed in terms of characters within the program (see the section on Addressing for details on character-relative references). Note that all entries in the SIZE column are considered by the Assembly System to be fixed in length. Therefore, any references to a field which is smaller than the figure given, or which follows such a field, can result in an improper address in the object program. Variable fields, then, should not be addressed by name nor should they appear before any field which is to be addressed by name.

Sequence numbers 2 through 6 on the accompanying Data Sheet illustrate the procedure for describing a file or work area which has been introduced by the ALOC function.

Sequence number 6 illustrates the use of the "REP." column. The area assigned to DEPOSITS will be 10 half-words long, with each of these half-words being accessible through a subscripting notation described in the section on Addressing.

The ALOC entry on line 7 illustrates the concept of overlaying a previously defined area. The entry in the "LEFT ADDRESS" column indicates that the area BC is to overlay part of area A. The LHE of area BC will be the 11th half-word in area A.

In allocating memory, the Assembly System maintains a series of counters so that the third allocation of memory (no. 13) will be placed immediately following the highest address assigned thus far. In this case, it will follow area A.

The use of the REN (Rename) function is illustrated on line 8. The entry in the "DATA NAME" column will receive its LHE from the current value of the memory counter and its RHE will be a function of the size of the entry. The memory counter is advanced in accordance with the units of the REN entry. Therefore, the next entry will receive the same LHE as the REN entry. Thus, by referring to BCDATE (the name plus the file area prefix) the programmer may address six characters, which have been further sub-divided into MONTH, DAY, and YEAR, each of which is, in itself, directly addressable by name. It is therefore possible to sub-divide a given area into many different groupings by using a series of REN functions. These may occur not only at the beginning of the area description, but throughout the description of the sub-divisions. Note that the REN function requires that the units of the entry immediately following a REN line be less than or equal to the units of the REN line; i.e., if the REN line is defined in terms of W, the following line may be defined in terms of W, H, or C; if the REN line is defined in terms of H, the following line must be defined in terms of H or C; if the REN line is defined in terms of C, the following line must be defined in terms of C.

The DUP entry illustrated on line 13 indicates that the description of area A, named in the "BIT CODE" column, is to be copied and assigned to a new area T. As with an ALOC entry, a DUP may overlay another area through an entry in the "LEFT ADDRESS" column; however, the overlaid area must have been named on its ALOC line.

B. DESCRIBING CONSTANTS

The Data Sheet is used as follows to define Constants:

1. Sequence

The "SEQ." column is used in the same manner as it is in describing Data and Working Storage.

2. Data Name

Certain data names have special meanings in the definition of constants.

First, the name

CON-FIX

must appear on the line allocating the area for the Fixed Constants, i.e., those constants which will be in memory throughout the program. Similarly.

CON-SEGnn

must appear when memory is being allocated for the constants that are to appear with segment nn of the program. The value of nn can range from 1 to 99.

The definitions of constants must always be in the proper order. That is, CON-FIX must precede CON-SEG 1, etc. Although they must be in order, it is not mandatory that there be an allocation for every segment within the program. Nor is there a requirement that there be Fixed Constants in every program.

3. Function

The functions allowable in the definition of constants are the following:

ALOC - Allocate

REN - Rename

FCON - Floating Point Constants

4. Size

Under "NO." the size of individual constants are given in terms of the units specified under "U". The Repeat function may be used to describe constants of like formats.

5. Bit Code

An entry must appear in the "BIT CODE" column of every constant being defined. Note that this does not apply to ALOC Lines. On all but FCON lines, the bit code entry may be 3, 4, 6 or B. On FCON lines, the entry may be D or B.

6. Left Address

This column is not available for the description of constants.

7. Description

The constant being defined is expressed in this column.

C. RULES FOR DEFINITION OF CONSTANTS

1. ALOC Lines

ALOC lines for constant description:

a. Must have a name in the form CON-FIX or CON-SEGnn in the "DATA NAME" column.

b. May not have an entry in the "NO.", "REP.", "LEFT ADDR." or "DESCRIPTIONS" columns.

c. May take a W or H entry in the "U" column, or it may be left blank. If H is used or it is left blank, the constant area will begin in the next available half-word without regard for word orientation. If W, the constant area will begin in the next available word location.

d. May take an entry of 3, 4, or 6 in the "BIT CODE" column.

e. The entry immediately following an ALOC line must be defined in units less than or equal to the units of the ALOC line.

2. REN Line

Rename Lines give the programmer the facility to give a common name to a group of constants. Rename lines;

a. Must have an entry in the "DATE NAME"; "NO", "U", and "BIT CODE" columns.

b. May have an entry in the "REP." column.

c. May not have an entry in the "DESCRIPTION" and "LEFT ADDR" column.

d. The entry immediately following a REN line must be defined in units that are less than or equal to the units of the REN line.

3. FCON Lines

FCON lines are used to define floating point constants. FCON lines:

a. Must have an entry in the "DATA NAME", "NO.", "U", "BIT CODE" and "DESCRIPTION" column. A floating point constant may occupy one or two words.

b. May have an entry in the "REP." column.

c. May not have an entry in the "LEFT ADDR" column.

d. The entry in the units column must be W.

e. The entry in the "BIT CODE" column may be either D or B, where

D - indicates that the constant described in the "DESCRIPTION" column is to be converted into a decimal floating point number in 4 bit code.

B - indicates that the constant described in the "DESCRIPTION" column is to be converted into a binary floating point number.

Floating point constants may be specified in the "DESCRIPTION" column in any of the following formats:

±XXXX

±XX.XX

±XXXXE±n

±XX.XXE±n

A number may be written without a sign. In this case the sign is considered to be plus. If, however, a sign appears as the first character of the description it is considered to be the sign of the floated number. The constant, itself, may contain a decimal point which will be taken into consideration in floating the number. In addition, an exponent may be specified with the constant. This is done by following the number with an E, followed by the exponent itself. The exponent may be a one or two digit number and may be preceded by a sign. If no sign is written, the sign is assumed to be plus.

4. Bit Code Rules

An entry of 3, 4, 6 or B in "BIT CODE" of non-FCON constants will cause the constant described in the "DESCRIPTION" column to be converted to the designated code and placed in the next available area as specified by the "U" column.

Should the programmer specify less information in the "DESCRIPTION" column for a particular constant than indicated by the size field, then that information will be right justified within the specified size, with the remainder of the field filled with zeros of the specified bit code. All octal digits will be accepted for conversion to 3 bit code; all those characters representable in four-bit code will be accepted for conversion to four-bit code, with the exception of comma; and all RCA 601 six-bit characters will be accepted for conversion to six-bit code, with the exception of comma.

A B entry in "BIT CODE" will cause the decimal number entered in the "DESCRIPTION" column to be converted to a binary number. Only H or W may be used to describe the units of a B bit code constant. If H, the "NO." entry must be either 1 or 2. If W, the "NO." entry must be 1. If the B bit code entry is described as being one half-word long, and the constant is described with a sign, the sign is placed in the rightmost bit of the half-word. If the B bit code entry is described as being two half-words long and the constant is described with a sign, the sign is placed in the rightmost bit positions of both half-words. If the constant is described as being one word long and the constant is described with a sign, the sign is placed in the rightmost bit of the word.

In any case, if the constant is not described with a sign, the constant will be right justified in the specified size.

B bit-code constants are described in the "DESCRIPTION" column in the following format:

±XXX

±XXXXE±n

±XXXXPn

±XX.XXPn

±XXXXE±nPn

±XX.XXE±nPn

If the leftmost character is a sign it is considered to be the sign of the number. If the leftmost character is not a sign, the number is considered to be unsigned. If the number does not contain a decimal point, and no P notation is given, the binary point will be considered to be to the right of the rightmost bit of the generated binary number. If a decimal point is given in the constant to be converted, a P notation must be given, i.e., a P followed by a one or two digit number indicating how many places to the right the binary point should be removed from the PØ position. PØ designates that the binary point is just to the left of the most significant bit of the field. In addition, an E notation may be used to describe the number where E may be followed by a one or two digit exponent preceded by a sign. If no sign precedes the exponent, it is considered to be positive. The presence or absence of a sign in the exponent does not affect the signed or unsigned condition of the number.

5. Repeating Constants

To use the "REP" column for constants, the number of constants of like format to be described are entered under "REP". They are then described in the "DESCRIPTION" column, separated by commas.

The accompanying Data Sheet illustrates the various procedures for defining constants. Lines 1 through 5 on the sample Data Sheet illustrate the procedure for allocating memory for Fixed Constants.

Line 2 does not contain an entry in the "FUNCT" column since it is not allocating memory, it is merely defining the constant to be generated. It calls for an 11 character constant which is found in the "DESCRIPTION" column. The characters are 6 bits each.

The next line shows a similar constant. However, the generated characters are only 4 bits in length. Since the preceding definition did not end in such a position as to allow an immediate assignment of a 4 bit character, the Assembly System will determine the next legitimate starting place for the constant, leaving zeros in the two unused bit positions (19 and 20).

The succeeding definition calls for a full half-word to be devoted to the constant appearing in the "DESCRIPTION" column. Since it is smaller than the area specified in the "SIZE" column, it will be right-justified within the area, with zeros filling any unused positions. It should be noted that this constant will be placed in the next available full half-word following the constant specified on line 3.

A series of three bit characters is defined on line 5.

DATA SHEET

601 ASSEMBLY SYSTEM

NAME ______________________ PAGE ________ OF ________

SEQ.	DATA NAME	FUNCT.	SIZE			BIT	LEFT ADDR.		DESCRIPTION
			NO.	U	REP.	CODE	HALF W	A	
1	CON-FIX	ALOC		W					
2	PRINT-1		11	C		6			CHANGE-PAGE
3	PRINT-2		10	C		4			123456789 +
4	CONST-1		1	H		6			21
5	CONST-2		8	C		3			01253257
6	CON-SEG1	ALOC		W					
7	NUMBER-1	FCON	1	W		D			+2.54
8	NUMBER-2	FCON	1	W	3	D			21.4, 157, +521.75
9	NUMBER-3	FCON	1	W		D			⊖ 4.15
10	NUMBER-4	FCON	2	W		D			+154E⊖ 2
11	PRINTOUT		3	W		6			SEGMENT-1-IN
12	CON-SEG2	ALOC		W					
13	FACTOR-1	FCON	1	W		B			⊖ 15.4
14	FACTOR-2	FCON	2	W	2	B			⊖ 254E ⊖ 2, 125E ⊖ 1
15	CON-SEG3	ALOC							
16	INCRMTS		1	H	5	3			0, 10, 100, 1000, 10000
17	CON-SEG3	ALOC							
18	NUMBER-5		1	H	3	B			125, +1.25P3, ⊖ 1.25E ⊖ 5P0

The next set of definitions describe the constants for segment 1 of the program. They are mostly floating-point constants. The number appearing in the "DESCRIPTION" column will be converted into a proper floating-point constant occupying 1 or 2 words, as specified in the "SIZE" column of each entry. The characteristic of the floating-point number is determined in either of two ways. First, if the decimal point is given in the description of the constant, the characteristic will be adjusted accordingly. If it is not present, it is assumed to be at the right-hand-end of the number being described. An examination is then made to determine if an exponent (E) has been specified. If it has, the decimal point will be shifted left or right according to the sign following the E. A plus sign (or the absence of a sign) will shift the point to the right, while a minus sign will shift it to the left. Every floating-point constant will receive a sign, regardless of whether the programmer has supplied a sign in his description. If none is given, a plus sign is supplied. The sign of the number, if given, is the leftmost character of its description.

The "BIT CODE" entry, B, in line 13 indicates that the floating-binary equivalent of ⊖ 15.4 is desired.

Lines 8 and 14 illustrate a procedure for listing a series of floating-point quantities. The number of entries in the list (up to 9999) is placed in the "REP" column. Thus, three words will be allotted for the entries on line 8 and four words for those on line 14. Note that the entries are separated by commas.

Similarly, lines 16 and 18 illustrate a procedure for listing a series of 3 bit constants and a series of binary constants, respectively.

IV. ADDRESSING

A. FILE AREAS AND WORKING STORAGE

All subdivision names appearing on the DATA SHEET are directly addressable by the name appearing in the "DATA NAME" column. However, they must be prefixed by the name assigned to the area in which they appear if that area was assigned a name in the ALOC line. For example, if NAME appears in area A, references to it must be to ANAME. This then allows the programmer to have duplicate symbolics in different areas and still be able to address them directly. If, however, NAME appears in an unnamed area, reference to it must be to NAME alone.

The area itself may be addressed by using just the letters found in the "DATA NAME" column of the ALOC or DUP entry, if such an entry was made, i.e., area BC may be addressed directly as BC. If no name was assigned to the area, it may not be addressed directly.

References may also be made in terms of the following relative notations:

1. + nnnn (0-9999)

   The programmer may reference a particular element of an area defined with the REP (Repeat) function by using the + nnnn notation. For example, if ENTRY has been described as being one word long, but repeated 10 times, then 10 contiguous words of HSM will be reserved within the data area. References to ENTRY will generate the address of the first word of that area. References to ENTRY + 5, however, will generate the address of the 6th word of the area. Note that the absence of a subscript notation is identical to + 0, i.e., they both refer to the first element of the area.

2. (± nnnn) (0-9999)

   Use of this form of reference allows the programmer to address a particular word, half-word or character within a defined area. The particular unit involved is determined by the definition of the area. For example, if TAX is defined in terms of words, TAX (10) would refer to the 11th word within the area defined by TAX. If it were defined by characters, TAX (10) would refer to the 11th character, etc. The absence of a sign preceding nnnn is taken to mean that the value is positive. A minus sign, however, means that the element desired precedes the referenced area. For example, TAX (-10) refers to the 10th word preceeding TAX.

3. (L ± nnnn) or (R ± nnnn) (0-9999)

   Particular characters within an area may be referenced by this notation, where L refers to the LHE of the area and R to the RHE.

All of these forms of addressing may be used in combination with each other. However, certain rules of hierarchy must be observed. The order of combination is:

1. NAME
2. + nnnn
3. (± nnnn)
4. (L ± nnnn) or (R ± nnnn)

In addition to the NAME, any or all of the relative notations may appear. They are all considered as part of the symbolic name which may then have Address Modifier and Indirect Address indicators appended to it.

For example:

* ENTRY + 3 (4), M5, I5

might be an address that would appear in the "ADDRESS or CONDITION" column of the program sheet.

B. CONSTANTS

Constants, like File and Working Storage, may be referenced using the + nnnn, (± nnnn) and (L ± nnnn) or (R ± nnnn) notations for relative addressing.

C. PROGRAM

Instructions are referenced relative to the symbolic appearing in the NAME column of the program sheet. In addition, the following relative notations are available:

1. + nnnn.nn

   Since not all instructions need be named, the programmer must refer to the unnamed ones with relative addresses. This particular notation provides access to unnamed instructions by relating them to the last named one.

   Thus,

   SORT + 1

   refers to the first instruction following the instruction called SORT. The .nn notation deals with inserted instructions as explained under the discussion of the program sheet in Chapter II.

2. (± nnnn)

   Use of this notation allows the programmer to access any half-word within an instruction, a RSRV area, or a LIST. The use of (0) produces the operation half-word or the first half-word in the RSRV or LIST area. A minus sign preceding nnnn gives rise to the address of a particular half-word preceding the referenced instruction.

3. (L ± nnnn) or (R ± nnnn)

   This notation allows access to a particular 3 bit character within a half-word.

   For example:

   SORT + 1 (2) (L + 2)

   would generate the address of the third character within the second address half-word of the instruction following SORT.

The order of combination of program references is:

1. NAME
2. + nnnn.nn
3. (± nnnn)
4. (L ± nnnn) or (R ± nnnn)

As with the other forms of addressing, these symbolics may be followed by Address Modifier or Indirect Address indicators on the program sheet.

V. SUBROUTINES

The RCA 601 Assembly System has been designed to permit the use of a library of subroutines. These subroutines are stored in pseudocode format on a library tape, and most of the features of the Assembly System are available to the subroutine programmer; i.e., subroutines may make use of certain Assembly System verbs and have their own data sheets for the description of constants and working storage.

Basically, there are two forms of subroutines that may be written for the 601 Assembly System. They are OPEN and CLOSED subroutines. An OPEN subroutine is one that is executed in sequence with the instructions of the calling routine. A CLOSED subroutine, however, is one which may be entered from several places in the calling routine and which will ultimately return control to the instructions following the exit from the calling routine. In order to enter a CLOSED subroutine, the programmer must use a JUMP AND STORE instruction.

The programmer may call a subroutine from the library through the use of an inclusion line. The entry in the NAME column of the inclusion line is the name by which the programmer of the calling routine wishes to refer to the subroutine. In many cases this will be identical to the name of the subroutine; however, if it is desired to have more than one copy of the same subroutine included in the coding of the object program, this will not be the case. As an example, if it is desired to include three different versions of the SINE subroutine, the entries in the NAME column might be SINA, SINB, and TRIGFUNCT. The first entry in the ADDRESS column is the name of the subroutine desired, such as SINE or ENCODE.

Parameters may be supplied either at the time the subroutines are inserted into the psuedocode or at the time they are executed in the object program. If, while writing a subroutine, the programmer wishes to operate upon a parameter which must be supplied by the user, he may specify that the parameter may be found in a particular address in the inclusion line. He does this by writing %nn in place of the required parameter. The characters nn may range from 1 to 99 and refer to the 1st to 99th address of the inclusion line. The user then follows the name of the subroutine with its parameters. Each of these must be preceded with an asterisk. For example, if the SINE subroutine required the location of the argument Z (LOCZ) and accuracy to which it should be computed (ACCZ) as parameters, the inclusion line might be as follows:

```
                    SUBR                    *SINE*LOCZ*ACCZ
```

The Assembly System will automatically substitute the symbolic LOCZ for every address within the subroutine which originally contained %1 and ACCZ for those which contained %2.

In addition, a closed subroutine may obtain "dynamic" parameters from the half-words following the JUMP AND STORE instruction which was used to enter the routine. These may be picked up at the time of execution by referring to them relative to the address of the JUMP AND STORE instruction. It is the responsibility of the subroutine programmer to save this address for such references and to generate the return JUMP to the calling routine. The parameters may be entered into a LIST verb following the JUMP AND STORE instruction.

Each symbolic in the NAME and ADDRESS columns within a subroutine is automatically prefixed with a specified symbol to avoid a duplication of names between the subroutine and the calling routine. Since each subroutine should receive a different prefix, all communications between subroutines, and with the calling routine, except for dynamic parameters, must be through the addresses appearing in the inclusion lines.

It should be noted that the line immediately following an inclusion line in any routine must be assigned a name and a sequence number. Thus, if an inclusion line has the name SINZ, and the succeeding line in the calling routine is given the name STEP2, the latter may be referenced only by using the symbolic name STEP2, not SINZ + 1. Reference to SINZ will yield the address of the first instruction, of the SINE Subroutine, SINZ + 1 the second instruction, etc.

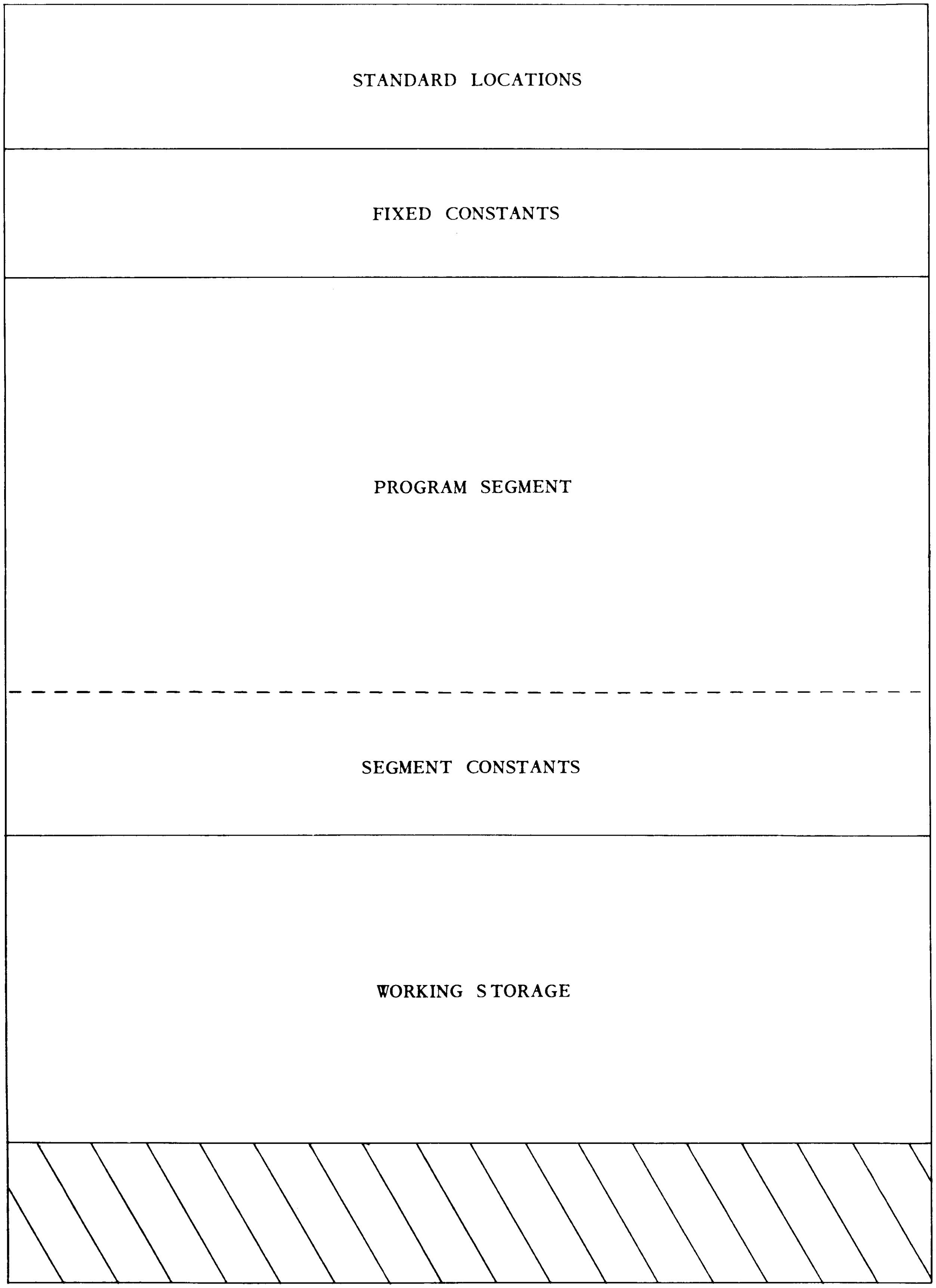
The Assembly System Subroutine Library, which appears on the Assembly Library tape, is maintained through the use of a special service routine supplied with the system. This routine, in addition to providing various edits, allows the programmer to define new subroutines and modify or delete previously-defined ones.

Additional information on the writing and use of subroutines will be published as an addendum to this manual.

VI. AUTOMATIC MEMORY LAYOUT

Memory for the object program will be laid out in the following manner:

LOWER LIMIT



The LOWER LIMIT of memory may be assigned by the programmer at the time of Assembly. However, each object program is produced in a relocatable format so that a new LHE may be specified at the time it is loaded into memory for execution. The RHE of the area occupied by the program and working storage is variable according to the requirements of the program. That is, working storage is assigned following the longest segment of the program or following any particular segment, as specified through the DEFW Descriptor Verb. Each program segment consists of the instructions and constants appearing in one block of the object program.

VII. OPERATION

A. CORRECTIONS

Corrections may be applied to the input to the RCA 601 Assembly System either during initial assembly or reassembly. Provisions are made to insert, delete or replace any number of instructions, constants or data definitions.

B. INPUT

Input to the Assembly System is from either cards, magnetic tape, or paper tape, In addition to the instructions and Descriptor Verbs, it includes a "description" of the computer being used to assemble the program, and the computer on which the program will be run.

Although instructions need not be written in the proper sequence for initial assembly, the storage and constant definitions must be entered in the proper order. That is, Fixed Constants must be defined before those for segment one, etc. Memory will be assigned for storage in the order in which it is defined in the input.

C. OUTPUT

As output, the Assembly System produces an object program on magnetic tape in proper PLT format. In addition, certain printouts are produced for the on-line printer. These include:

1. A listing of the constants and storage allocations made by the programmer and the associated memory assignments made by the Assembly System.

2. A cross-reference between the ordered pseudocode and the generated machine code.

3. A complete listing of all format errors detected by the Assembly System during the assembly process. These errors are retained on magnetic tape until the assembly has been completed and some form of object program has been produced. The programmer may then decide whether to reassemble with corrections or to debug the program as produced.

In addition, all information necessary for reassembly will be retained on magnetic tape.

Complete operating instructions will be available in a subsequent publication.

VIII. DESCRIPTOR VERBS

Certain operation codes have been provided with the RCA 601 Assembly System to allow the programmer to further describe various aspects of his program to the Assembly System. These Descriptor Verbs are identified by their operation codes and thus, may appear at any position throughout the program. However, they must each have a name and sequence number, as must the instruction following each of them. Comments may not appear on Descriptor Verbs. The Verbs are as follows:

A. NAME

One NAME verb must appear in every program. It takes only one address, the name that the programmer has assigned to this program. This name must be alphanumeric and may not exceed 32 characters in length.

B. RSRV

The single address for this verb indicates the number (0-9999) of half-words to be reserved in the place where the RSRV appears. The first half-word to be reserved is assigned the name on the RSRV line. However, references to the RHE will yield the RHE of the entire area. In this case, only half-word and character-relative addresses are applicable in referencing the half-words in the reserved area.

C. LIST

The purpose of the LIST verb is to create a series of half-words containing the machine code definitions of various symbolics such as instruction or data addresses. The generated list (not to exceed 4 addresses per pseudo-code line, nor 55 lines of pseudo-code) may also be used for the storage of certain constants, e.g., switch settings, masks, indirect addresses, etc. The symbolics to be defined are entered in the address portions of the LIST verb, with each entry being preceded by an asterisk (*). Each entry may then be referred to relative to the name given to the LIST verb by the programmer by using the half-word and character-relative notations. The name, itself, refers to the first or last half-word in the list according to the requirements of the instruction in which it is used.

D. MACH

By the use of the operation code, MACH, the programmer may write sets of machine coding, by entering a series of half-words, each consisting of 9 octal digits, as the addresses of this instruction. The 9 octal digits represent the tag bits and the 24 information bits of the half-word. In selecting tags, one should consider the information given under Machine Code Addresses in Chapter II of this manual. As many as 4 addresses, each preceded by an asterisk (*), may be written on one pseudocode line, and as many as 55 lines may be used for one MACH verb. Addressing is identical to that for the LIST and RSRV verbs.

E. EQUAL

The EQUAL verb requires two addresses. The first address contains a symbolic which does not appear in the "NAME" column of the coding or data sheets, but is to be assigned to the same memory location as the name appearing in the second address. The latter must be a symbolic that appears in the "NAME" column of the coding or data sheet.

F. SGMT

The SGMT verb requires three addresses. The first address indicates the sequence number of the first instruction of the segment of coding being defined. The second address may take one of three forms:

1. *The sequence number of the first instruction of the area that will be overlaid by the segment.*

2. *The special symbolic, ESGMTJ, where J is a number from 1 through 98. This indicates that the segment being defined is to begin immediately following the constants for the segment whose number is J.*

3. *A machine address in the form, #TIHHHHHC, where the #TIHHHHHC represents the machine location into which the programmer wishes the LHE of this segment to be placed.*

The third address is the segment number of the segment being defined. The number may range from two through 99. Note that segment 1 need not be defined. The first segment is considered to begin with the first line of pseudo-code in sorted order.

If a sequence number is specified in the second address, that number must be less than the sequence number specified in the first address. If the ESGMTJ option is used in the second address J must be less than the number specified in the 3rd address.

Examples:

```
SGMT      *25*10*2
SGMT      *100*ESGMT1*3
SGMT      *250*#00123450*4
```

G. DEFW

The programmer may, by use of this verb, override the automatic assignment of the LHE of the work area. This instruction takes only one address in the form ESGMTJ, where J specifies the segment number following which the work area is to begin. The LHE of the work area then becomes the next available word or half-word following the end of segment J.

H. RINS

This verb will cause the generation of coding in the object program to effect the loading of a new program segment. The generated coding provides certain parameters to a standard insertion routine which locates the required program block, inserts it, and transfers control to an address specified in the RINS verb. The two addresses required by this verb are *N*RETURN, where N is a number from 2 through 99 specifying the segment being called for, and RETURN is the address to which control is to be returned. RETURN may be a symbolic or machine address.

I. PD

This verb specifies the type of equipment that each relative peripheral device in the program is to address. The verb may have as many addresses as are necessary to meet the program requirements. PD verb addresses are written in the following format:

$$*TEn_1n_1, n_2n_2, n_3n_3, n_4n_4 \ldots n_nn_n$$

where TE is the type equipment being defined and may equal:

MT(33)	for a 33 KC magnetic tape station
MT (66)	for a 66 KC magnetic tape station
MT (120)	for a 120 KC magnetic tape station
CR	for a Card Reader
CP	for Card Punch
PTR	for a Bulk Paper Tape Reader
SR	for the Strip Paper Tape Reader
PTP	for Paper Tape Punch
OLP	for On Line Printer
MP	for the Monitor Printer

n_1n_1 through n_nn_n are those relative peripheral device numbers that address the device named in TE.

No more than 13 relative numbers may be named in any one address. Should a program use more than 13 peripheral devices of the same type a second address with the same TE portion can be written. All peripheral devices used in the program must be defined by a PD verb. More than one PD verb can be used in any one program, though, in general, one should suffice.

Example:

The PD verb:

```
PD *MT(120) 01, 02, 03 *SR10 *MP77
```

indicates that relative peripheral device numbers 01, 02, and 03 are to address 120 KC Magnetic Tape Stations; 10 is to address the Strip Paper Tape Reader; and 77 the Monitor Printer.

IX. ASSEMBLY SYSTEM ORDER CODE APPENDICES

The accompanying charts show the standard format for each instruction in the RCA 601 Assembly System. Presented below is a legend of the symbols used on the charts.

A. GENERAL DESCRIPTION OF ORDER CODE CHARTS

1. Instruction

The name of the instruction being described is found in this column.

2. Operation

The symbolic operation codes presented in this column have been grouped into generalized categories with variation characters following the broken line on the charts. Both the category name and the variation characters (if any) must appear on the program sheet for each instruction.

3. T

An X appears in this column if the T column on the program sheet may be used.

4. NI

An X appears in this column if the NI column on the program sheet may be used.

5. COUNT

The range of the numeric value that may be placed in this column on the program sheet for each instruction may be found here.

6. COND.

The conditions that may be entered as the first entry in the Address or Condition column of the program sheet (no asterisk precedes conditions) for each instruction may be found here. Symbolics used for CONDITIONS are as follows:

U - for half-word arithmetics, specifies that unsigned arithmetic is to be performed.

REGISTER - those registers that may be affected by an instruction are listed under this heading.

SS,US,SU or UU - for the symbol field controlled arithmetics, one of these four notations is used. The first letter refers to the first operand; the second letter to the second operand. S = signed and U = unsigned.

4 or 6 - in the Convert Code instruction indicates the code to which conversion is to be made.

CONNECTIVE - in the FORM instructions, a connective code of 0 through 7 must be entered.

S - for the Monitor instruction, S specifies store after staticizing.

H - for the Read Blocks Forward instruction, H specifies that the count refers to half-words rather than blocks. This option is used when reading gapless paper tape.

D - for the Rewind instruction, D specifies a rewind and disconnect

P,N,Z - for the Sense PRI instruction P, N, and Z may be entered singly or in any combination. When in combination they are entered with no separators, i.e.,

P

PN

PZ

PNZ

where P specifies PRP

N specifies PRN

Z specifies PRZ

ALL - in the Shift instructions, ALL specifies a shift including sign.

E - in the WRITE instructions, E specifies Erase.

7. AD_1, AD_2, AD_3, AD_4

The standard addresses for each instruction are shown in these columns. Each instruction must be written with the number of asterisks shown, or with 3 for non-input-output instructions, and 4 for input-output instructions with the exception of those instructions noted in the N column.

A line through a column means that this column on the program sheet must be left blank.

8. N

The numbers in this column indicate notes applying to this instruction.

1. No addresses, other than those indicated on the charts, may be specified with these instructions.

2. No operation half-word is generated for these instructions.

3, 4, 5, 6. In general, the addresses given on the charts are, in order, the A, B, C addresses and the Peripheral Device Half-Word respectively. The following numbers in the N column, however, indicate exceptions and how they vary from the general instruction format.

3. AD_1 specifies the C address

4. AD_1 specifies the C address
   AD_2 specifies the PD half-word

5. AD_1 specifies the B address
   AD_2 specifies the C address
   AD_3 specifies the PD half-word

6. AD_1 specifies the PD half-word.

ORDER CODE CHART

INSTRUCTION	OPERATION		T	NI	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
ADD WORD DECIMAL	ADD	WD	X	X	———	———	*WORD ADDRESS OF AUGEND OR & (MAY BE ABSOLUTE)	*WORD ADDRESS OF ADDEND OR & (MAY BE ABSOLUTE)	*WORD ADDRESS OF SUM OR & (MAY BE ABSOLUTE)	———	
ADD WORD BINARY	ADD	WB	X	X	———	———	*WORD ADDRESS OF AUGEND OR & (MAY BE ABSOLUTE)	*WORD ADDRESS OF ADDEND OR & (MAY BE ABSOLUTE)	*WORD ADDRESS OF SUM OR & (MAY BE ABSOLUTE)	———	
ADD HALF-WORD DECIMAL	ADD	HD	X	X	1 or 2	U OR BLANK	*RHE OF AUGEND OR &	*RHE OF ADDEND OR &	*RHE OF SUM OR &	———	
ADD HALF-WORD BINARY	ADD	HB	X	X	1 or 2	U OR BLANK	*RHE OF AUGEND OR &	*RHE OF ADDEND OR &	*RHE OF SUM OR &	———	
ADD SYMBOL/FIELD CONTROLLED	ADD	S	X	X	1 to 64	SS, US, UU, or SU	*LHE OF AUGEND	*LHE OF ADDEND	*LHE OF SUM	———	
ADD CHARACTER ADDRESS	ADD	CA	X	X	———	———	*ADDRESS OF AUGEND	*ACTUAL VALUE OF ADDEND or INDIRECT ADDRESS THEREOF	*ADDRESS OF RESULT HALF-WORD	———	
ADD HALF-WORD ADDRESS	ADD	HA	X	X	———	———	*ADDRESS OF AUGEND	*ACTUAL VALUE OF ADDRESS or INDIRECT ADDRESS THEREOF	*ADDRESS OF RESULT HALF-WORD	———	
ADD TO ADDRESS REGISTER	ADD	AR	X	X	———	*REG-ISTER* AAR, BAR, ICC, ECC, TAR or SIC	*ACTUAL VALUE OF INCREMENT or INDIRECT ADDRESS THEREOF			———	3

ORDER CODE CHART (Cont'd)

INSTRUCTION	OPERATION		T	NI	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
CONVERT CODE	CONVT	CD	X	X	———	4 or 6	* LHE OF AREA TO BE CON- VERTED	* RHE OF AREA TO BE CON- VERTED	* LHE OF DES- TINATION AREA	———	
DIVIDE WORD DECIMAL	DIV	WD	X	X	1 or 2	———	* RHE OF DIVIDEND OR & (MAY BE ABSOLUTE)	* ADDRESS OF DIVISOR OR & (MAY BE ABSOLUTE)	* ADDRESS OF QUOTIENT OR & (MAY BE ABSOLUTE)	———	
DIVIDE HALF- WORD DECIMAL	DIV	HD	X	X	1 or 2	U OR BLANK	* RHE OF DIVIDEND OR &	* ADDRESS OF DIVISOR OR &	* ADDRESS OF QUOTIENT OR &	———	
DIVIDE HALF- WORD BINARY	DIV	HB	X	X	———	———	* ADDRESS OF DIVIDEND OR &	* ADDRESS OF DIVISOR OR &	* ADDRESS OF QUOTIENT OR &	———	
DIVIDE SYMBOL/ FIELD CONTROLLED	DIV	S	X	X	1 to 64	SS, US, SU, or UU	* RHE OF DIVIDEND	* RHE OF DIVISOR	* LHE OF QUOTIENT	———	
DO	DO		X	X	1 to 256	———				———	
EDIT FIELD LEFT TO RIGHT	EDIT	FL	X	X	1 to 256	———	* LHE OF FIELD	* LHE OF MASK	* LHE OF RESULT	———	
EDIT FIELD RIGHT TO LEFT	EDIT	FR	X	X	1 to 256	———	* RHE OF FIELD	* RHE OF MASK	* RHE OF RESULT	———	
ENABLE JUMP	ENABLE		———	X	———	———	* PERIPHERAL DEVICE HALF- WORD				6
FILL	FILL		X	X	———	———	* LHE OF FILL AREA	* RHE OF FILL AREA	* SYMBOL HALF WORD	———	

ORDER CODE CHART (Cont'd)

INSTRUCTION	OPERATION		T	N I	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
FIND SYMBOL LEFT TO RIGHT	FIND	SL	X	X	1 to 256	—	* LHE OF SEARCH AREA	* RHE OF SEARCH AREA	* HALF-WORD ADDRESS OF FINAL SETTING STORE	—	
FIND SYMBOL RIGHT TO LEFT	FIND	SR	X	X	1 to 256	—	* LHE OF SEARCH AREA	* RHE OF SEARCH AREA	* HALF-WORD ADDRESS OF FINAL SETTING STORE	—	
FIND ABSENCE OF SYMBOL LEFT TO RIGHT	FIND	ASL	X	X	—	—	* LHE OF SEARCH AREA	* RHE OF SEARCH AREA		—	
FIND ABSENCE OF SYMBOL RIGHT TO LEFT	FIND	ASR	X	X	—	—	* LHE OF SEARCH AREA	* RHE OF SEARCH AREA		—	
FLOAT DECIMAL	FLOAT	D	X	X	—	—	* ADDRESS OF CHARACTER-ISTIC			—	
FLOAT BINARY	FLOAT	B	X	X	—	—	* ADDRESS OF CHARACTER-ISTIC			—	
FORM CHARACTERS A	FORM	CA	X	X	1 to 64	*CONNEC-TIVE* 0 TO 7	* RHE OF OPERAND 1	* RHE OF OPERAND 2	* RHE OF DES-TINATION AREA	—	
FORM CHARACTERS B	FORM	CB	X	X	1 to 64	*CONNEC-TIVE* 0 TO 7	* RHE OF OPERAND 1	* RHE OF OPERAND 2	* RHE OF DES-TINATION AREA	—	
FORM HALF-WORD A	FORM	HA	X	X	—	*CONNEC-TIVE* 0 TO 7	* HALF-WORD ADDRESS OF OPERAND 1 OR &	* HALF-WORD ADDRESS OF OPERAND 2 OR &	* HALF-WORD ADDRESS OF DESTINATION AREA OR &	—	

ORDER CODE CHART (Cont'd)

INSTRUCTION	OPERATION		T	N I	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
FORM HALF-WORD B	FORM	HB	X	X	—	*CONNEC-TIVE* 0 TO 7	* HALF-WORD ADDRESS OF OPERAND 1 OR &	* HALF-WORD ADDRESS OF OPERAND 2 OR &	* HALF-WORD ADDRESS OF DESTINATION AREA OR &	—	
FORM WORDS A	FORM	WA	X	X	—	*CONNEC-TIVE* 0 TO 7	* WORD ADDRESS OF OPERAND 1 OR &	* WORD ADDRESS OF OPERAND 2 OR &	* WORD ADDRESS OF DESTINATION AREA OR &	—	
FORM WORDS B	FORM	WB	X	X	—	*CONNEC-TIVE* 0 TO 7	* WORD ADDRESS OF OPERAND 1 OR &	* WORD ADDRESS OF OPERAND 2 OR &	* WORD ADDRESS OF DESTINATION AREA OR &	—	
HALT AND WAIT	HALT		—	—	—	—	—	—	—	—	1
INSERT TAG	INSERT	T	X	X	1 to 256		* LHE OF AREA INTO WHICH TAGS ARE TO BE INSERTED			—	
INSERT BITS	INSERT	B	X	X	—	—	* HALF-WORD ADDRESS OF OPERAND OR &	* HALF-WORD ADDRESS OF EXTRACT PATTERN OR &	* HALF-WORD ADDRESS OF RESULT OR &	—	
JUMP	JUMP		—	—	—	—	*JUMP ADDRESS	—	—	—	1,2
JUMP IF SET	JUMP	S	—	—	—	—	* JUMP ADDRESS	—	—	—	1,2
JUMP IF RESET	JUMP	R	—	—	—	—	* JUMP ADDRESS	—	—	—	1,2
JUMP AND STORE	JUMP	AS	—	—	—	—	* JUMP ADDRESS			—	
JUMP IF SET AND STORE	JUMP	SAS	—	—	—	—	* JUMP ADDRESS			—	
JUMP IF RESET AND STORE	JUMP	RAS	—	—	—	—	* JUMP ADDRESS			—	

ORDER CODE CHART (Cont'd)

INSTRUCTION	OPERATION		T	NI	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
LOAD MACHINE AND JUMP	LOAD		—	—	—	—	* WORD ADDRESS DESIGNITATING LHE OF INFOR-MATION TO BE LOADED			—	
MONITOR	MONIT		X	X	—	S OR BLANK				—	
MOVE BY CHARACTER LEFT TO RIGHT	MOVE	CL	X	X	—	—	* LHE OF ORIGIN AREA	* RHE OF ORIGIN AREA	* LHE OF DES-TINATION AREA	—	
MOVE BY HALF-WORDS LEFT TO RIGHT	MOVE	HL	X	X	—	—	* LHE OF ORIGIN AREA	* RHE OF ORIGIN AREA	* LHE OF DES-TINATION AREA	—	
MOVE BY WORD LEFT TO RIGHT	MOVE	WL	X	X	—	—	* LHE OF ORIGIN AREA	* RHE OF ORIGIN AREA	* LHE OF DESTINATION AREA	—	
MOVE BY CHARACTER RIGHT TO LEFT	MOVE	CR	X	X	—	—	* LHE OF ORIGIN AREA	* RHE OF ORIGIN AREA	* RHE OF DESTINATION AREA	—	
MOVE BY HALF-WORDS RIGHT TO LEFT	MOVE	HR	X	X	—	—	* LHE OF ORIGIN AREA	* RHE OF ORIGIN AREA	* RHE OF DESTINATION AREA	—	
MOVE BY WORD RIGHT TO LEFT	MOVE	WR	X	X	—	—	* LHE OF ORIGIN AREA	* RHE OF ORIGIN AREA	* RHE OF DESTINATION AREA	—	
MOVE FIELD BY CHAR-ACTER LEFT TO RIGHT	MOVE	FCL	X	X	1 to 256	—	* LHE OF ORIGIN AREA	* LHE OF DESTINATION FIELD		—	

ORDER CODE CHART (Cont'd)

INSTRUCTION	OPERATION	T	NI	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
MOVE FIELD BY HALF-WORD LEFT TO RIGHT	MOVE FHL	X	X	1 to 256	—	* LHE OF ORIGIN AREA	* LHE OF DESTINATION FIELD		—	
MOVE FIELD BY WORD LEFT TO RIGHT	MOVE FWL	X	X	1 to 256	—	* LHE OF ORIGIN AREA	* LHE OF DESTINATION FIELD		—	
MOVE FIELD BY CHAR-ACTER RIGHT TO LEFT	MOVE FCR	X	X	1 to 256	—	* RHE OF ORIGIN AREA	* RHE OF DESTINATION FIELD		—	
MOVE FIELD BY HALF-WORD RIGHT TO LEFT	MOVE FHR	X	X	1 to 256	—	* RHE OF ORIGIN FIELD	* RHE OF DESTINATION FIELD		—	
MOVE FIELD BY WORD RIGHT TO LEFT	MOVE FWR	X	X	1 to 256	—	* RHE OF ORIGIN FIELD	* RHE OF DESTINATION FIELD		—	
MOVE SYMBOL CONTROLLED BY CHAR-ACTER LEFT TO RIGHT	MOVE SCL	X	X	1 to 256	—	* LHE OF ORIGIN ITEM	* LHE OF DESTINATION AREA		—	
MOVE SYMBOL CONTROLLED BY CHAR-ACTER RIGHT TO LEFT	MOVE SCR	X	X	1 to 256	—	* RHE OF ORIGIN ITEM	* RHE OF DESTINATION AREA		—	
MOVE TAG	MOVE T	X	X	1 to 256	—	* HALF-WORD ADDRESS OF WORD WHOSE TAG IS TO BE MOVED	* FIRST ADDRESS INTO WHICH TAG IS TO BE PLACED		—	

ORDER CODE CHART (Cont'd)

INSTRUCTION	OPERATION	T	NI	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
MOVE ADDRESS	MOVE A	X	X	—	—	* ACTUAL VALUE TO BE MOVED OR INDIRECT ADDRESS THEREOF	* ADDRESS OF HALF-WORD INTO WHICH VALUE IS TO BE PLACED		—	
MULTIPLY AND ACCUMLATE WORD DECIMAL	MAC WD	X	X	—	—	* ADDRESS OF MULTIPLICAND OR & (MAY BE ABSOLUTE)	* ADDRESS OF MULTIPLIER OR & (MAY BE ABSOLUTE)	* RHE OF RESULT OR & (MAY BE ABSOLUTE)	—	
MULTIPLY AND ACCUMULATE HALF-WORD DECIMAL	MAC HD	X	X	—	U OR BLANK	* ADDRESS OF MULTIPLICAND OR &	* ADDRESS OF MULTIPLIER OR &	* RHE OF PRODUCT OR &	—	
MULTIPLY WORD DECIMAL	MULT WD	X	X	—	—	* ADDRESS OF MULTIPLICAND OR & (MAY BE ABSOLUTE)	* ADDRESS OF MULTIPLIER OR & (MAY BE ABSOLUTE)	* RHE OF PRODUCT OR & (MAY BE ABSOLUTE)	—	
MULTIPLY HALF-WORD DECIMAL	MULT HD	X	X	1 or 2	U OR BLANK	* RHE OF MULTIPLICAND OR &	* RHE OF MULTIPLIER OR &	* RHE OF PRODUCT OR &	—	
MULTIPLY HALF-WORD BINARY	MULT HB	X	X	—	—	* ADDRFSS OF MULTIPLICAND OR &	* ADDRESS OF MULTIPLIER OR &	* RHE OF PRODUCT OR &	—	
MULTIPLY SYMBOL/ FIELD CONTROLLED	MULT S	X	X	1 to 64	SS, SU, UU, OR US	* RHE OF MULTIPLICAND	* RHE OF MULTIPLIER	* RHE OF PRODUCT	—	
READ BLOCKS FORWARD	READ BF	X	X	1 to 256	H OR BLANK	* ADDRESS IN WHICH TO PLACE FIRST HALF-WORD READ	* ADDRESS BEYOND WHICH NO DATA MAY BE READ INTO MEMORY	* HALF-WORD ADDRESS OF LHE OF FINAL SETTINGS (MAY BE GUARDED)	* PERIPHERAL DEVICE HALF-WORD	

ORDER CODE CHART (Cont'd)

INSTRUCTION	OPERATION		T	NI	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
READ SYMBOLS FORWARD	READ	SF	X	X	1 to 256	———	* ADDRESS IN WHICH TO PLACE FIRST HALF-WORD READ	* ADDRESS BEYOND WHICH NO DATA MAY BE READ INTO MEMORY	* HALF-WORD ADDRESS OF LHE OF FINAL SETTINGS (MAY BE GUARDED)	* PERIPHERAL DEVICE HALF-WORD	
READ BLOCK FORWARD GENERATE LIST	READ	BFG	X	X	———	———	* ADDRESS IN WHICH TO PLACE FIRST HALF-WORD READ	* ADDRESS BEYOND WHICH NO DATA MAY BE READ INTO MEMORY. LHE MINUS ONE OF LIST	* HALF-WORD ADDRESS OF LHE OF FINAL SETTINGS (MAY BE GUARDED)	* PERIPHERAL DEVICE HALF-WORD	
READ BLOCKS REVERSE	READ	BR	X	X	1 to 256	———	* ADDRESS IN WHICH TO PLACE FIRST HALF-WORD READ	* ADDRESS BEYOND WHICH NO DATA MAY BE READ INTO MEMORY	* HALF-WORD ADDRESS OF LHE OF FINAL SETTINGS (MAY BE GUARDED)	* PERIPHERAL DEVICE HALF-WORD	
READ SYMBOLS REVERSE	READ	SR	X	X	1 to 256	———	* ADDRESS IN WHICH TO PLACE FIRST HALF-WORD READ	* ADDRESS BEYOND WHICH NO DATA MAY BE READ INTO MEMORY	* HALF-WORD ADDRESS OF LHE OF FINAL SETTINGS (MAY BE GUARDED	* PERIPHERAL DEVICE HALF-WORD	
READ BLOCK REVERSE GENERATE LIST	READ	BRG	X	X	———	———	* ADDRESS IN WHICH TO PLACE FIRST HALF-WORD READ	* ADDRESS BEYOND WHICH NO INFORMATION IS TO BE READ INTO MEMORY. RHE PLUS ONE OF LIST	* HALF-WORD ADDRESS OF LHE OF FINAL SETTINGS (MAY BE GUARDED)	* PERIPHERAL DEVICE HALF-WORD	

ORDER CODE CHART (Cont'd)

INSTRUCTION	OPERATION		T	NI	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
REWIND	RWD		—	X	—	D OR BLANK	* HALF-WORD ADDRESS OF LHE OF FINAL SETTING (MAY BE GUARDED)	* PERIPHERAL DEVICE HALF-WORD			4
SENSE PRI	SENSE	PRI	—	X	—	P, N, Z IN ANY COMBIN-ATION				—	
SENSE AND	SENSE	AND	—	X	—	—	* HALF-WORD ADDRESS OF MASK			—	
SENSE OR	SENSE	OR	—	X	—	—	* HALF-WORD ADDRESS OF MASK			—	
SET ADDRESS REGISTER	SET	AR	X	X	—	*REGIS-TER* LR, AAR, BAR, CAR, ICC, ECC, or TAR	* ACTUAL VALUE TO BE PLACED IN REGISTER OR INDIRECT ADDRESS THEREOF			—	
SET DATA REGISTER	SET	DR	X	X	—	*REGIS-TER* CR, CLR, PIR, MIR, ETC, IR, PIB, RI, MWA, LWA, MHA, LHA, SYR or TR	* ADDRESS OF HALF-WORD OR WORD WHOSE CON-TENTS WILL BE PLACED IN SPECIFIED REGISTER			—	
SHIFT ACCUMLATOR BITS TO THE RIGHT	SHIFT	ABR	—	X	1 to 64	ALL OR BLANK				—	

ORDER CODE CHART (Cont'd)

INSTRUCTION	OPERATION		T	NI	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
SHIFT ACCUMULATOR CHARACTERS TO THE RIGHT	SHIFT	ACR	—	X	1 to 64	ALL OR BLANK				—	
SHIFT ACCUMULATOR BITS TO THE LEFT	SHIFT	ABL	—	X	1 to 64	ALL OR BLANK				—	
SHIFT ACCUMULATOR CHARACTERS TO THE LEFT	SHIFT	ACL	—	X	1 to 64	ALL OR BLANK				—	
SHIFT MOST SIGNIFICANT ACCUMULATOR BITS TO THE RIGHT	SHIFT	MBR	—	X	1 to 64	ALL OR BLANK				—	
SHIFT MOST SIGNIFICANT ACCUMULATOR CHARACTERS TO THE RIGHT	SHIFT	MCR	—	X	1 to 64	ALL OR BLANK				—	
SHIFT MOST SIGNIFICANT ACCUMULATOR BITS TO THE LEFT	SHIFT	MBL	—	X	1 to 64	ALL OR BLANK				—	
SHIFT MOST SIGNIFICANT ACCUMULATOR CHARACTERS TO THE LEFT	SHIFT	MCL	—	X	1 to 64	ALL OR BLANK				—	

ORDER CODE CHART (Cont'd)

INSTRUCTION	OPERATION		T	NI	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
RING SHIFT MOST SIGNIFICANT ACCUMULATOR TO THE RIGHT	SHIFT	R	—	X	1 to 64	—				—	
SHIFT HALF-WORD BITS TO THE RIGHT	SHIFT	HBR	X	X	1 to 64	ALL OR BLANK	* ADDRESS OF HALF-WORD TO BE SHIFTED OR &			—	
SHIFT HALF-WORD BITS TO THE LEFT	SHIFT	HBL	X	X	1 to 64	ALL OR BLANK	* ADDRESS OF HALF-WORD TO BE SHIFTED OR &			—	
SHIFT HALF-WORD ACCUMULATOR CHARACTERS TO THE RIGHT	SHIFT	HCR	—	X	1 to 64	ALL OR BLANK				—	
SHIFT HALF-WORD ACCUMULATOR CHARACTERS TO THE LEFT	SHIFT	HCL	—	X	1 to 64	ALL OR BLANK				—	
SHIFT DOUBLE HALF-WORD ACCUMULATOR BITS TO THE RIGHT	SHIFT	DBR	—	X	1 to 64	—				—	
SHIFT DOUBLE HALF-WORD ACCUMULATOR CHARACTERS TO THE RIGHT	SHIFT	DCR	—	X	1 to 64	—				—	

ORDER CODE CHART (Cont'd)

INSTRUCTION	OPERATION		T	NI	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
SHIFT DOUBLE HALF-WORD ACCUMULATOR BITS TO THE LEFT	SHIFT	DBL	——	X	1 to 64	——				——	
SHIFT DOUBLE HALF-WORD ACCUMULATOR CHARACTERS TO THE LEFT	SHIFT	DCL	——	X	1 to 64	——				——	
STORE ADDRESS REGISTER	STORE	AR	X	X	——	*REGIS-TER* LR, AAR, BAR, CAR, ICC, ECC or TAR	* ADDRESS OF HALF-WORD INTO WHICH REGISTER IS TO BE STORED			——	3
STORE AND LOAD MACHINE	STORE	LM	——	——	——	——	* WORD ADDRESS DESIGNATING LHE OF STORAGE AREA			——	
STORE INPUT-OUTPUT AND JUMP	STORE	IO	X	X	——	——	* HALF-WORD ADDRESS OF LHE OF FINAL SETTINGS (MAY BE GUARDED)	* PERIPHERAL DEVICE HALF-WORD			4
STORE DATA REGISTER	STORE	DR	X	X	——	*REGIS-TER* MIR, PIR, IR, MWA, LWA, CR, SYR, TR, MHA or LHA	* ADDRESS OF HALF-WORD OR WORD WHERE SPECIFIED REGISTER IS TO BE STORED			——	3

ORDER CODE CHART (Cont'd)

INSTRUCTION	OPERATION		T	NI	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
SUBTRACT WORD DECIMAL	SUB	WD	X	X	—	—	* ADDRESS OF MINUEND OR & (MAY BE ABSOLUTE)	* ADDRESS OF SUBTRAHEND OR & (MAY BE ABSOLUTE)	* ADDRESS OF DIFFERENCE OR & (MAY BE ABSOLUTE)	—	
SUBTRACT WORD BINARY	SUB	WB	X	X	—	—	* ADDRESS OF MINUEND OR & (MAY BE ABSOLUTE)	* ADDRESS OF SUBTRAHEND OR & (MAY BE ABSOLUTE)	* ADDRESS OF DIFFERENCE OR & (MAY BE ABSOLUTE)	—	
SUBTRACT HALF-WORD DECIMAL	SUB	HD	X	X	1 or 2	U OR BLANK	* RHE OF MINUEND OR &	* RHE OF SUBTRAHEND OR &	* RHE OF DIFFERENCE OR &	—	
SUBTRACT HALF-WORD BINARY	SUB	HB	X	X	1 or 2	U OR BLANK	* RHE OF MINUEND OR &	* RHE OF SUBTRAHEND OR &	* RHE OF DIFFERENCE OR &	—	
SUBTRACT SYMBOL/ FIELD CONTROLLED	SUB	S	X	X	1 to 64	SS, US, UU, or SU	* RHE OF MINUEND	* RHE OF SUBTRAHEND	* RHE OF DIFFERENCE	—	
SUBTRACT CHARACTER ADDRESS	SUB	CA	X	X	—	—	* ADDRESS OF MINUEND HALF-WORD	* ACTUAL VALUE OF SUBTRAHEND OR INDIRECT ADDRESS THERE OF	* ADDRESS OF RESULT HALF-WORD	—	
SUBTRACT HALF-WORD ADDRESS	SUB	HA	X	X	—	—	* ADDRESS OF MINUEND HALF-WORD	* ACTUAL VALUE OF SUBTRAHEND OR INDIRECT ADDRESS THERE OF	* ADDRESS OF RESULT HALF-WORD	—	
SWAP HALF-WORDS	SWAP	H	X	X	—	—	* ADDRESS OF HALF-WORD TO BE SWAPPED	* ADDRESS OF OTHER HALF-WORD TO BE SWAPPED		—	

ORDER CODE CHART (Cont'd)

INSTRUCTION	OPERATION		T	NI	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
SWAP WORDS	SWAP	W	X	X	—	—	* ADDRESS OF WORD TO BE SWAPPED	* ADDRESS OF OTHER WORD TO BE SWAPPED		—	
SWAP ADDRESS REGISTER	SWAP	AR	X	X	—	*REGISTER* LR, AAR, BAR, CAR, ICC, ECC or TAR	* ADDRESS OF HALF–WORD TO BE SWAPPED			—	
SWAP DATA REGISTER	SWAP	DR	X	X	—	*REGISTER* MIR, PIR, IR, CR, SYR, MHA, LHA, TR	* ADDRESS OF HALF–WORD WHOSE CONTENTS ARE TO BE INTERCHANGED WITH SPECIFIED REGISTER			—	
TALLY	TALLY		X	X	—	—	* HALF–WORD ADDRESS OF TALLY QUANTITY			—	
TEST BY CHARACTER LEFT TO RIGHT	TEST	CL	X	X	1 to 256	—	* LHE OF MINUEND	* LHE OF SUBTRAHEND		—	
TEST BY CHARACTER RIGHT TO LEFT	TEST	CR	X	X	1 to 256	—	* RHE OF MINUEND	* RHE OF SUBTRAHEND		—	
TEST BY HALF–WORD	TEST	H	X	X	1 or 2	—	* LHE OF MINUEND OR &	* LHE OF SUBTRAHEND OR &		—	

ORDER CODE CHART (Cont'd)

INSTRUCTION	OPERATION		T	NI	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
TEST WORD	TEST	W	X	X	—	—	* ADDRESS OF MINUEND OR &	* ADDRESS OF SUBTRAHEND OR &		—	
WIND SENTINELS FORWARD	WIND	SF	X	X	1 to 256	—	* HALF-WORD ADDRESS OF LHE OF FINAL SETTINGS (MAY BE GUARDED)	* PERIPHERAL DEVICE HALF-WORD			4
WIND GAPS FORWARD	WIND	GF	—	X	1 to 256	—	* HALF-WORD ADDRESS OF LHE OF FINAL SETTINGS (MAY BE GUARDED)	* PERIPHERAL DEVICE HALF-WORD			4
WIND SENTINELS REVERSE	WIND	SR	X	X	1 to 256	—	* HALF-WORD ADDRESS OF LHE OF FINAL SETTINGS (MAY BE GUARDED	* PERIPHERAL DEVICE HALF-WORD			4
WIND GAPS REVERSE	WIND	GR	—	X	1 to 256	—	* HALF-WORD ADDRESS OF LHE OF FINAL SETTINGS (MAY BE GUARDED)	* PERIPHERAL DEVICE HALF-WORD			4
WRITE BLOCK	WRITE	B	X	X	BLANK OR 1 to 64 FOR PRINTER	E OR BLANK	* ADDRESS OF LHE OF AREA TO BE WRITTEN	* ADDRESS OF RHE OF AREA TO BE WRITTEN	* HALF-WORD ADDRESS OF LHE OF FINAL SETTINGS (MAY BE GAURDED)	* PERIPHERAL DEVICE HALF-WORD	
WRITE BLOCK FROM LIST	WRITE	BL	X	X	1 to 256	E OR BLANK	* ADDRESS OF LHE OF LIST	* HALF-WORD ADDRESS OF LHE OF FINAL SETTINGS (MAY BE GUARDED)	* PERIPHERAL DEVICE HALF-WORD		5

ORDER CODE CHART (Cont'd)

INSTRUCTION	OPERATION		T	NI	COUNT	COND.	AD_1	AD_2	AD_3	AD_4	N
WRITE SYMBOLS	WRITE	S	X	X	1 to 256	E OR BLANK	* ADDRESS OF LHE OF AREA TO BE WRITTEN	* ADDRESS OF RHE BEYOND WHICH NO INFORMATION MAY BE WRITTEN	* HALF-WORD ADDRESS OF LHE OF FINAL SETTINGS (MAY BE GUARDED)	* PERIPHERAL DEVICE HALF-WORD	
WRITE SYMBOLS FROM LIST	WRITE	SL	X	X	1 to 256	E OR BLANK	* ADDRESS OF RHE BEYOND WHICH NO INFORMATION MAY BE WRITTEN AND LHE OF LIST MINUS 1 HALF-WORD	* HALF-WORD ADDRESS OF LHE OF FINAL SETTINGS (MAY BE GUARDED)	* PERIPHERAL DEVICE HALF-WORD		5

APPENDIX I

STANDARD LOCATIONS

The following symbolics are provided for the addressing of standard locations:

*$M10	-	Address Modifier 10
*$M11	-	The Increment to AM10
*$M20	-	Address Modifier 20
*$M21	-	The Increment to AM20
*$M30	-	Address Modifier 30
*$M31	-	The Increment to AM30
*$M40	-	Address Modifier 40
*$M41	-	The Increment to AM40
*$M50	-	Address Modifier 50
*$M51	-	The Increment to AM50
*$M60	-	Address Modifier 60
*$M61	-	The Increment to AM60
*$M70	-	Address Modifier 70
*$M71	-	The Increment to AM70
*$VF1	-	Overflow Jump address
*$VF2	-	Subsidiary Overflow Jump address
*$UF1	-	Underflow Jump address
*$UF2	-	Subsidiary Underflow Jump address
*$ZDI	-	Zero Divisor Jump address
*$TE	-	Tag Equal in Origin Jump address
*$TU	-	Tag Unequal in Origin Jump Address
*$TD	-	Tag Unequal in Origin or Destination Jump address
*$LLA	-	Lower Limit Alarm Jump address
*$ULA	-	Upper Limit Alarm Jump address
*$PRN	-	PRN Jump address
*$PRZ	-	PRZ Jump address
*$PRP	-	PRP Jump address
*$NFI	-	Not Found Indicator Jump address
*$TCC	-	Transfer Control Condition Jump address
*$IWI	-	Instruction Waiting Indicator Jump address
*$PI1	-	Program Indicator 1 Jump address
*$P12	-	Program Indicator 2 Jump address
*$P13	-	Program Indicator 3 Jump address

*$MPE	-	Memory Parity Error Jump address
*$BTE	-	Bus Transfer Error Jump address
*$MON	-	Monitor Standard Locations
*$ELX	-	Edit Location X
*$ELY	-	Edit Location Y
*$PDT	-	Peripheral Device Table (LHE unless otherwise specified)
*$STICC	-	Relative ICC storage location

Note that standard location addresses may be suffixed by the relative notations (±nnnn) and (L/R ± nnnn). The (±nnnn) notation will always be considered to be increments of half-words. The (L/R ± nnnn) notation will always yield increments in terms of *3-bit* characters. In addition address modifier notations and indirect addressing notations may be applied to these locations.

APPENDIX II

MASK HALF-WORD SYMBOLS

The following symbolics may be used in the Mask Half-Word:

VF1	-	Overflow indicator bit
VF2	-	Subsidiary overflow indicator bit
UF1	-	Underflow indicator bit
UF2	-	Subsidiary underflow indicator bit
ZDI	-	Zero divisor indicator bit
TE	-	Tag equal in origin indicator bit
TU	-	Tag unequal in origin indicator bit
TD	-	Tag unequal in origin or destination indicator bit
LLA	-	Lower Limit Alarm indicator bit
ULA	-	Upper Limit Alarm indicator bit
PRN	-	PRN bit
PRZ	-	PRZ bit
PRP	-	PRP bit
NFI	-	Not found indicator bit
TCC	-	Transfer Control Condition indicator bit
IWI	-	Instruction Waiting indicator bit
PI1	-	Program Indicator 1 bit
PI2	-	Program Indicator 2 bit
PI3	-	Program Indicator 3 bit
ETI	-	Elapsed time indicator bit
EI	-	External Indicator bit
SIT	-	Simultaneous Instruction Termination bit
DO	-	Do indicator bit

APPENDIX III

SPECIAL HALF-WORD SYMBOLS

Option 1

*/ Rxx, Cn, Tnn, Sx

where:

/	- is the introductory symbol of a Special Half-Word
Rxx	- refers to the round indicators. The "R" is the signal for the round indicator. The xx may be either H, W, HW or blank. If H, the generated half-word may be used to set the Half-Word Round Indicator and reset the Word Round Indicator. If W, the generated half-word may be used to set the Word Round Indicator and reset Half-Word Round Indicator. If HW, the generated half-word may be used to set both Round Indicators, and if blank, both Round Indicators may be reset.
Cn	- refers to the Character Length Register where n may equal 3, 4, 6 or 8, representing the bit size to which the register is to be set.
Tnn	- refers to the tag register, where nn may be one or two digits. If one digit, it is a number from 0 through 7, representing the tag to be placed in the register. If two digits, the first is as described above and the second is a one, indication that the tag transfer option is desired.
Sx	- refers to the Symbol Register, where x is the symbol to be placed into the register. It may be any punchable 4 or 6 bit character with the exception of space, *, >, <, ;, quotes, colon and comma, or may be three octal digits surrounded by quotes. The second form must be used for all 3 or 8 bit characters and for the exceptions noted above.

Option 2

* / LR LLR, ULR

where:

/	- is the introductory symbol of a Special Half-Word
LR	- specifies that the Limit Register option of the Special Half-Word is desired.
LLR	- specifies the Lower Limit Register setting desired. It may be LLR, in which case the lower limit register setting of the program being assembled is automatically supplied by the assembly system, or it may be specified as a number sign (#) followed by 4 octal digits.
ULR	- specifies the Upper Limit Register setting desired. It may be ULR, in which case the upper limit register setting of the program being assembled is automatically supplied by the assembly system, or it may be specified by a number sign (#) followed by 4 octal digits.

If the number sign followed by four octal digits is used, the four octal digits represent the first 4 digits of the address to which the limit is to be set, where the first digit may be one or zero and the last digit must be zero or four. The least significant three digits of the address (not specified with this option) are assumed to be zeros.

Examples:

```
* / LR LLR, ULR
* / LR LLR, #1750
* / LR #0454, #1750
* / LR #0454, ULR
```

The symbolics of option 2 of the Special Half-Word must be written in order shown above.

APPENDIX IV

REGISTER ADDRESSING

As shown on the Order Code charts, various instructions may address several registers. The symbolic indicating which register the instruction is to address is given as a condition. The symbolic register designations are as follows:

CR	=	Control Register
SYR	=	Symbol Register
TR	=	Tag Register
CLR	=	Character Length Register
RI	=	Round Indicators
PIR	=	Program Indicator Register
MIR	=	Mask Indicator Register
ETC	=	Elapsed Time Clock
IR	=	Instruction Register
PIB	=	Program Indicator Bits
MHA	=	Most Significant Half-Word Accumulator
LHA	=	Least Significant Half-Word Accumulator
MWA	=	Most Significant Full Word Accumulator
LWA	=	Least Significant Full Word Accumulator
LR	=	Limit Register
AAR	=	A Address Register
BAR	=	B Address Register
CAR	=	C Address Register
ICC	=	Instruction Control Counter
ECC	=	Elementary Control Counter
TAR	=	Temporary Address Register
SIC	=	Simultaneous Instruction Counter

APPENDIX V

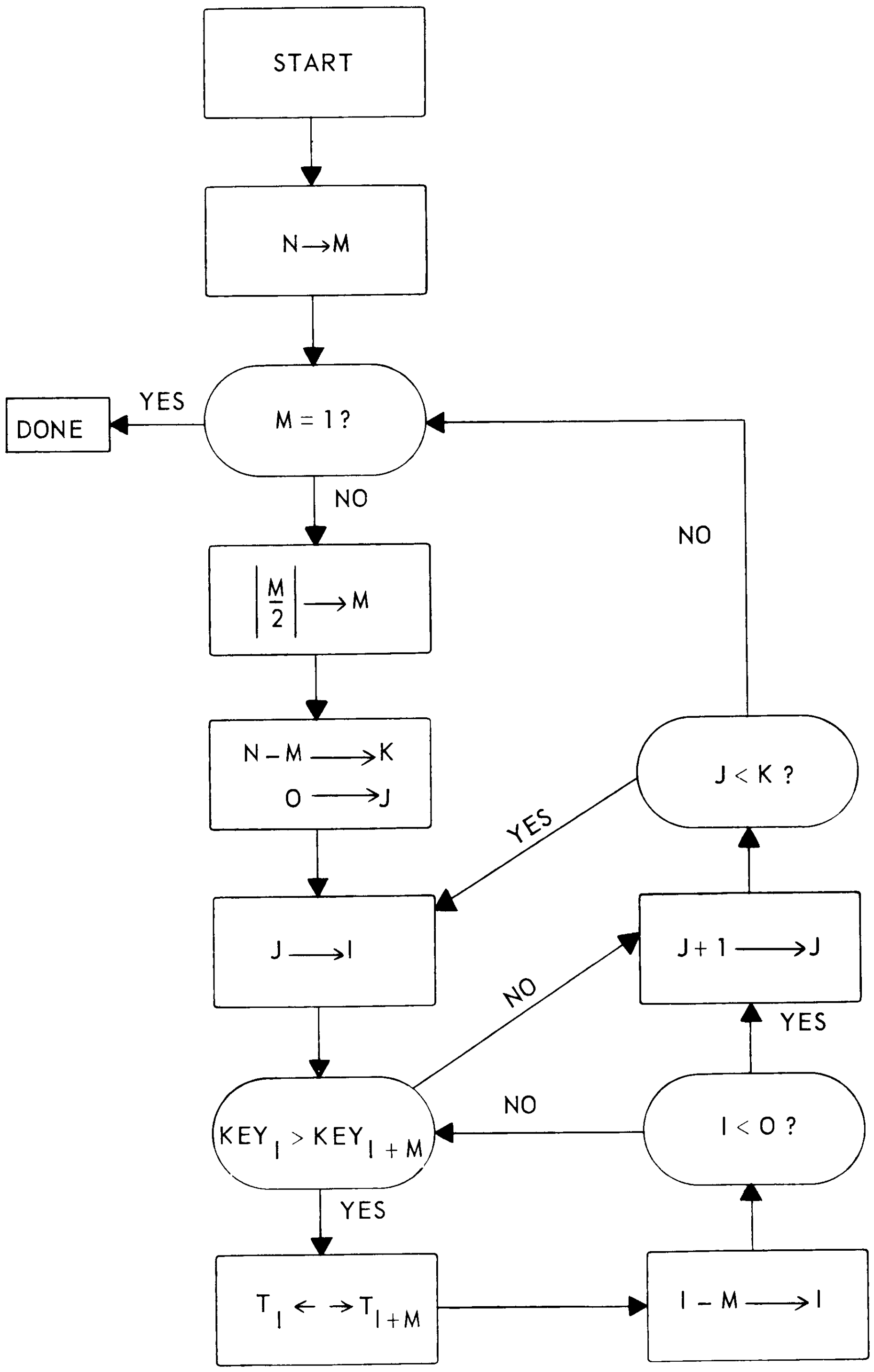
SAMPLE PROBLEM

The sample problem shown on the following pages, illustrating the use of the RCA 601 Assembly System, is a high-speed indirect internal sort. It is assumed that there are N consecutive one-word entries in a list. These entries address one-word keys which serve as the criteria. The keys may be randomly scattered in memory. The sort will order the entries in the list area according to the relative magnitudes of keys, such that, the entries in the list, reading from left to right, will address the keys in ascending sequence.

The problem assumes N (the number of entries in the list and the number of items to be sorted) to be 100. The list of addresses are in symbolic location TAU.

A. FLOW CHART

HIGH-SPEED SORT (SHELL)



B. DATA SHEET

NAME ______________

DATA SHEET
601 ASSEMBLY SYSTEM

PAGE 1
OF 1

HIGH SPEED SORT

SEQ.	DATA NAME	FUNCT.	SIZE			BIT CODE	LEFT ADDR.		DESCRIPTION
			NO.	U	REP.		HALF W	A	
1	CON - SEG 1	ALOC		H					
2	N		1	H		3			144
3	ZERO		1	H		3			Ø
4	ONE		1	H		3			2Ø
5		ALOC		H					
6	M		1	H					
7	J		1	H					
8	K		1	H					
9	TAU		1ØØ	W					
10	KEYS		1ØØ	W					

C. CODING

NAME ____________________

PROGRAM SHEET
601 ASSEMBLY SHEET

PAGE 1
OF 1

HIGH SPEED COST

SEQ.	NAME	OPERATION		T	NI	COUNT	ADDRESS or CONDITION
1	START	MOVE	FHL			1	*N*M: HIGH SPEED SORT
		SHIFT	HBL			4	ALL *M
2	OUTERLOOP	TEST	H			1	*M *ONE
		JUMP	S				*DONE
		SHIFT	HBR			4	*M
		SHIFT	HBL			3	*M
		ADD	HA				*MOD *M *$M2Ø
		SUB	HA				*ZERO *M *$M11
		SUB	HB			1	*N *M *K
		MOVE	FHL			1	*ZERO *J
3	INNERLOOP	MOVE	FHL			1	*J *$M1Ø
		TEST	W				*TAU, M1Ø, I* TAU, M2Ø, I
		SENSE	PRI				ZN
		JUMP	S				*OVER
		SWAP	W				*TAU, M20 *TAU, M11
		TEST	H			1	*$M1Ø * $M11
		SENSE	PRI				*N
		JUMP	S				*INNERLOOP + 1
4	OVER	ADD	HB			1	* J * ONE * J
		TEST	H			1	* J * K
		SENSE	PRI				* ZP
		JUMP	S				*OUTERLOOP
		JUMP					*INNERLOOP
5	MOD	LIST					* Ø, M1Ø

D. ASSEMBLY SYSTEM PRODUCED LISTINGS

```
                                                                                                  PAGE 001
                                  CONSTANTS AND WORKING STORAGE DESCRIPTIONS

SEQ.      DATA NAME       FUNCT.     SIZE     U     REP.     B     H. W.     A        LHE           RHE

00001     CON-SEG 1       ALOC                H                                     0011670

00002     N                           1       H                3                    0011670       0011677
  DESC:       144

00003     ZERO                        1       H                3                    0011700       0011707
  DESC:     0

00004     ONE                         1       H                3                    0011710       0011717
  DESC:       20

          AREA-RHE                                                                                0011717

00005                     ALOC                H                                     0011720

00006     M                           1       H                                     0011720       0011727

00007     J                           1       H                                     0011730       0011737

00008     K                           1       H                                     0011740       0011747

00009     TAU                       100       W                                     0011760       0015057

00010     KEYS                      100       W                                     0015060       0020157

          AREA-RHE                                                                                0020157
```

```
0001         START                                     HIGH SPEED SORT
                0011000         1 06000161        MOVEFHL       T:        NI:          COUNT: 1        COND:
                0011010         6 00011670                      (A)       * N
                0011020         6 00011720                      (B)       * M

             START + 1
                0011030         1 25000441        SHIFTHBL      T:        NI:          COUNT: 4        COND: ALL
                0011040         6 00011720                      (A)       * M

0002         OUTERLOOP
                0011050         1 27007162        TESTH         T:        NI:          COUNT: 1        COND:
                0011060         6 00011720                      (A)       * M
                0011070         6 00011710                      (B)       * ONE

             OUTERLOOP + 1
                                                  JUMPS         T:        NI:          COUNT:          COND:
                0011100         5 xxxxxxx4                                * DONE

             OUTERLOOP + 2
                0011110         1 24040441        SHIFTHBR      T:        NI:          COUNT: 4        COND:
                0011120         6 00011720                      (A)       * M

             OUTERLOOP + 3
                0011130         1 25040341        SHIFTHBL      T:        NI:          COUNT: 3        COND:
                0011140         6 00011720                      (A)       * M

             OUTERLOOP + 4
                0011150         1 62007071        ADDHA         T:        NI:          COUNT:          COND:
                0011160         6 00011660                      (A)       * MOD
                0011170         6 00011720                      (B)       * M
                0011200         6 00010040                      (C)       * $M20

             OUTERLOOP + 5
                0011210         1 63007071        SUBHA         T:        NI:          COUNT:          COND:
                0011220         6 00011700                      (A)       * ZERO
                0011230         6 00011720                      (B)       * M
                0011240         6 00010030                      (C)       * $M11
```

PROGRAM LISTING (Cont'd)

```
          OUTERLOOP + 6
             0011250        1 57007171      SUBHB        T:       NI:          COUNT: 1     COND:
             0011260        6 00011670                  (A)      * N
             0011270        6 00011720                  (B)      * M
             0011300        6 00011740                  (C)      * K

          OUTERLOOP + 7
             0011310        1 06000161      MOVEFHL      T:       NI:          COUNT: 1     COND:
             0011320        6 00011700                  (A)      * ZERO
             0011330        6 00011730                  (B)      * J

0003      INNERLOOP
             0011340        1 06000161      MOVEFHL      T:       NI:          COUNT: 1     COND:
             0011350        6 00011730                  (A)      * J
             0011360        6 00010020                  (B)      * $M10

          INNERLOOP + 1
             0011370        1 26007062      TESTW        T:       NI:          COUNT:       COND:
             0011400        6 12011760                  (A)     *TAU, M10, I
             0011410        6 22011760                  (B)     *TAU, M20, I

          INNERLOOP + 2
             0011420        1 41000602      SENSEPRI     T:       NI:          COUNT:       COND: Z N

          INNERLOOP + 3
                                            JUMPS        T:       NI:          COUNT:       COND:
             0011430        5 00011544                           * OVER

          INNERLOOP + 4
             0011440        1 00000062      SWAPW        T:       NI:          COUNT:       COND:
             0011450        6 20011760                  (A)      *TAU, M20
             0011460        6 14011760                  (B)      *TAU, M11

          INNERLOOP + 5
             0011470        1 27007162      TESTH        T:       NI:          COUNT: 1     COND:
             0011500        6 00010020                  (A)      * $M10
             0011510        6 00010030                  (B)      * $M11
          INNERLOOP + 6
            0011520        1 41000402      SENSEPRI      T:        NI:           COUNT:         COND: N

          INNERLOOP + 7
                                           JUMPS         T:        NI:           COUNT:         COND:
            0011530        5 00011374                          *INNERLOOP + 1

0004      OVER
            0011540        1 56007171      ADDHB         T:        NI:           COUNT: 1       COND:
            0011550        6 00011730                   (A)      * J
            0011560        6 00011710                   (B)      * ONE
            0011570        6 00011730                   (C)      * J

          OVER + 1
            0011600        1 27007162      TESTH         T:        NI:           COUNT: 1       COND:
            0011610        6 00011730                   (A)      * J
            0011620        6 00011740                   (B)      * K

          OVER + 2
            0011630        1 41000302      SENSEPRI      T:        NI:           COUNT:         COND: Z P

          OVER + 3
                                           JUMPS         T:        NI:           COUNT:         COND:
            0011640        5 00011054                          *OUTERLOOP

          OVER + 4
                                           JUMPS         T:        NI:           COUNT:         COND:
            0011650        5 00011343                          *INNERLOOP

0005      MOD
                                           LIST          T:        NI:           COUNT:         COND:
             0011660       2 10000000                        * 0,M10
```