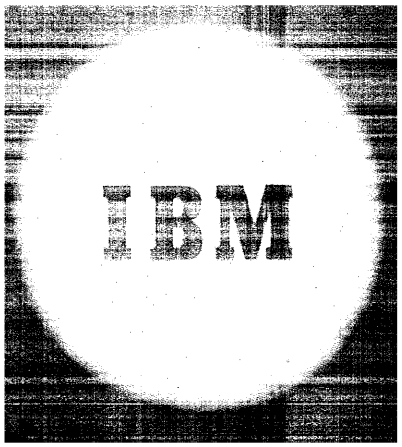

705

autocoder system

---

manual of information

MAJOR REVISION (February, 1957)

This edition, Form 22-6726-1, obsoletes Form 22-6726-0 and all earlier editions. Significant changes have been made throughout the manual, and this new edition should be reviewed in its entirety.

Form 22-6726-1

CONTENTS

THE 705 AUTOCODER SYSTEM

THE AUTOCODER is a simplified system of program writing for the 705 Data Processing Machine. This system relieves the programmer of considerable clerical work and removes many of the restrictions placed upon him by the machine-instruction coding system. Programs written in autocoder form can be combined and relocated at will because they are independent of actual memory location. Additions, corrections, or deletions to an autocoder program can be made by a reassembly process without changing the addresses written.

The autocoder recognizes either descriptive names of instructions or the 705 machine codes. Programs may be written with mnemonic operation codes, or with a mixture of mnemonic and single-character operation codes. The programmer may refer to data or instructions by name rather than by absolute memory location. The name is usually a mnemonic abbreviation which becomes the "tag" of the reference point. With tags, the program may be written with minimum comment and without reference to tables of locations for the data to which instructions refer.

The autocoder system incorporates all the 705 instructions plus an expanded set of autocoder special operations and macro-instructions. (Macro-instructions are special instructions developed by the autocoder programmer that permit one instruction to take the place of a number of 705 instructions.) In addition, one autocoder instruction or macro-instruction will incorporate from a library tape each subroutine which is called for in a particular program. Any program, or any part of a program, may easily be placed on the library tape as a subroutine. Combined, these features mean that the programming required for any specific job need be done only once.

An important point in handling business data is the ease of making changes in the form and contents of a record. A program written in autocoder form can be reassembled by the autocoder program with a new arrangement of any, or all, of the record formats and will operate exactly as it did originally. This feature may be extended. If the definitions of records common to an installation and used in several programs are entered as subroutines on the library tape, a change in the subroutine will permit machine correction of *all* programs which refer to these common records by the autocoder program reassembly.

With the autocoder, each field has a unique designation. The autocoder program adjusts references to this field as required by various instructions which refer to the right or left position of the field, or the left location plus four (as in high-speed receive and transmit). In the autocoder system it is not necessary for the programmer to give every instruction a "tag." Only those instructions which are referred to by other instructions in the program need to be tagged. Tags not only eliminate considerable writing, but also serve to identify the critical instructions which are operated upon or to which transfers are made. This simplifies checking a program after it has been written.

One useful feature of the autocoder is the use of literal operands. One entry can specify an operation and arrange for the storage in memory of the constant or coefficient which is the actual data to be used in that operation. The autocoder program picks up this constant from the literal operand, assigns it a location, and inserts that address in the program. Constants specified as literal operands are stored only once in memory, and subsequent occurrences of the same literal operand will reuse the same stored constant.

After a program is written and punched, in the first pass the autocoder program:

1. Assigns memory locations to all record and constant areas and fields.
2. Assigns memory locations to all tagged and untagged instructions.
3. Translates mnemonic operation codes to the machine operation code.
4. Incorporates the 705 instructions corresponding to each macro-instruction used.
5. Incorporates subroutines as required.
6. Compiles tag tables for all tagged locations.

On the second pass, the autocoder program:

1. Determines (from the tag tables) and adjusts the actual machine addresses of all operands.

2. Punches directly, or through auxiliary operations, a machine program complete with a loading program and transfers from the loading program to the first instruction written.
3. Prints directly, or through auxiliary operations, a listing of the original program exactly as written and the resulting machine program.

Even the best programmers sometimes make mistakes, but the likelihood of error is greatly reduced because the autocoder will remember field locations and lengths and do most of the clerical-type work rapidly and accurately. Meaningful abbreviations will help the programmer keep things straight. Macro-instructions and subroutines will be already checked, free from error.

In addition, the autocoder will detect some mistakes and will warn the programmer through the typewriter. For example, because multiplication must be carried out in the accumulator, the autocoder will correct any attempt to multiply in an auxiliary storage unit. The program is expected to be in sequence, and if entry cards are not in order, a message will be typed. However, because he was combining different programs or for other reasons, the programmer may have planned to have this occur; therefore, no correction is attempted. No mistake the programmer can make will stop the assembly. Each error encountered by the autocoder program is listed on the typewriter, so that all errors may be dealt with at one time.

The information presented in this manual should be adequate to furnish a working knowledge of the autocoder system. A thorough understanding of the 705 *Preliminary Manual of Operation* and of basic programming techniques is assumed.

ORGANIZATION

ALL information for a 705 program is originally supplied by the programmer. The information is written in autocoder language on the program sheet (Figure 1). Machine instructions, record areas, work areas, and all other program data are described in complete detail. Each separate item of information forming a program entry is written on one line of a sheet.

Entries are written in the order in which they are to be stored in the memory of the 705, unless some other sequence is specifically indicated by the programmer. A space is provided for writing program page numbers. Line numbers are preprinted on the sheet.

After the program has been written and checked, it is key punched into autocoder instruction cards (Figure 2). Each card contains one program entry. Card field arrangement conforms exactly with the columnar arrangement of the program sheet. All information on each line is punched exactly as written, including the page and line number. An identification of the entire program may be gang-punched into all cards.

Punching is verified for accuracy by the card verifier or by checking the original sheet with listed or interpreted cards. After verification, the instruction cards are sorted to line number, then to page number, to place the program entries in the exact sequence given by the programmer.

The instruction cards may then be used as direct input to the 705, or cards may first be converted to tape records to provide a faster input to the program assembly (Figure 3). In either case, the program

PAGE 01

PROGRAM SALARY PAYROLL DATE DEC 30 IDENT. PAYRL 1

LINE	TAG	OPERATION	NUM	OPERAND	COMMENTS
010	GROSS TO NET	SEL		202	Read input
020		RD		PAY RECORD	
030		TRS		END OF JOB	
040		TRA		READ ERROR	
050		RAD	1	(+0000)	Set and fill ASUs
060		RAD	2	(+000)	
070		RAD		TAX CLASS	Compute exemption amount
080		MPY		(-1300)	
090		ADD		GROSS PAY	
100		TRZ		GTN 1	Test for withholding tax
110		TRP		GTN 2	
120	GTN 1	ST	1	W/H TAX	No withholding - store zero
130		TR		GTN 3	
140	GTN 2	MPY		(+18)	Compute W.H. Tax

FIGURE 1. ORIGINAL PROGRAM

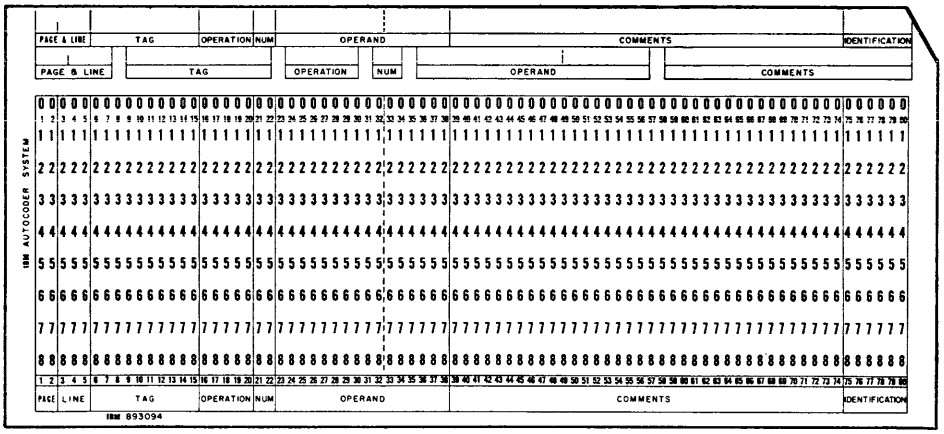

FIGURE 2. INSTRUCTION CARD

instructions are one input to the 705; the autocoder system tape is the second.

The autocoder system tape (Figure 4) contains several types of information:

1. *System control.* Instructions to load the autocoder into the 705 memory and miscellaneous instructions to control the form of input and output as directed by the console operator.

2. *Librarian.* Instructions to control the revision of the library.

3. *Phase I.* Instructions and controls of the first phase of assembly.

4. *Phase II.* Instructions and controls of the second phase of assembly.

5. *Macro-instructions.* A library of tested sequences of 705 instructions. Each sequence is identified on the tape by a name.

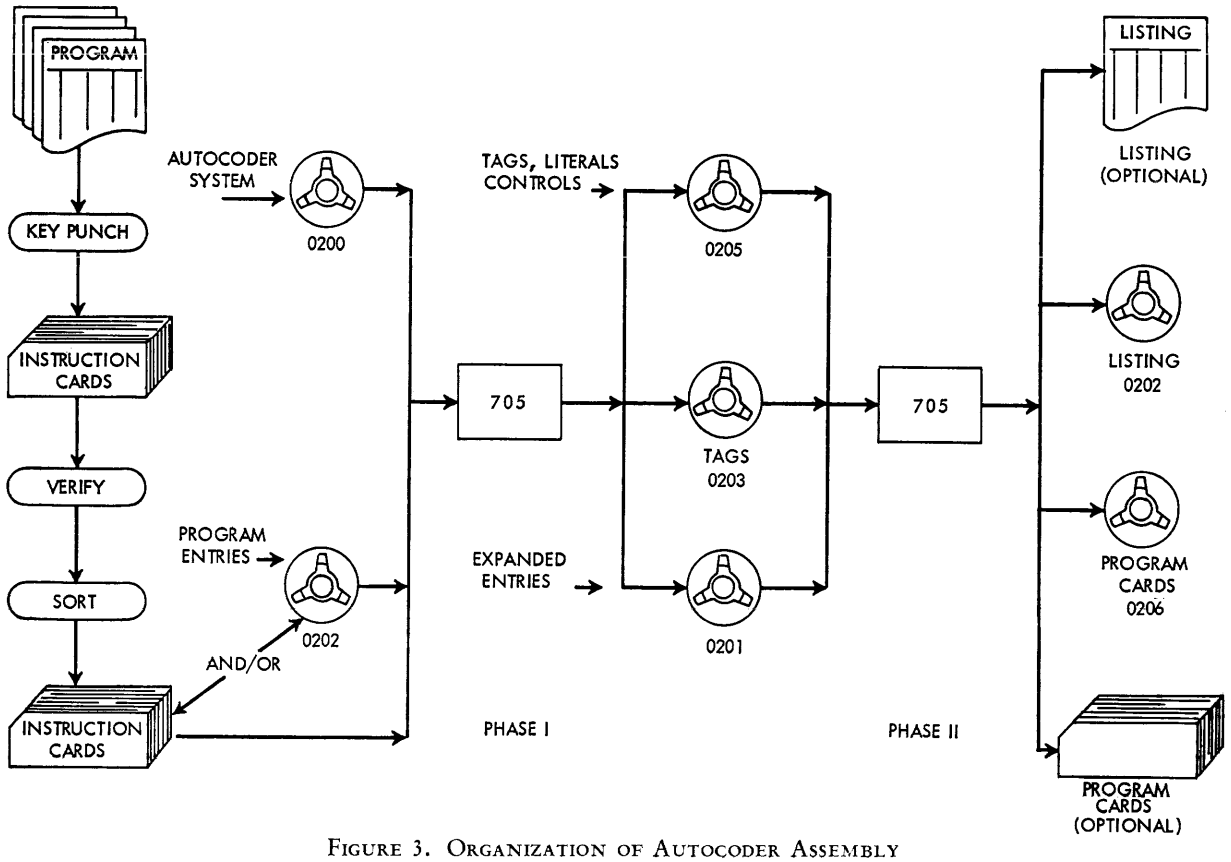

FIGURE 3. ORGANIZATION OF AUTOCODER ASSEMBLY

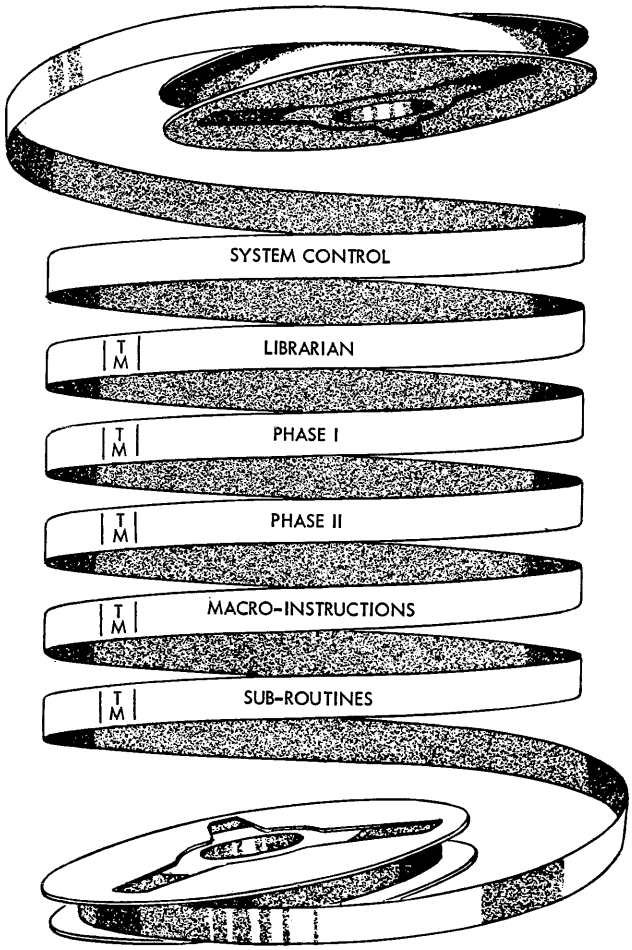

FIGURE 4. AUTOCODER SYSTEM TAPE

6. *Subroutines.* A library of tested 705 subroutines. Each routine is also identified on the tape by its name.

During assembly, the 705 operates upon the instruction records, one at a time, as input data. The entire assembly is under control of the autocoder tape. A complete 705 program is produced as output. Instructions originally written in autocoder language are automatically converted into five-character 705 absolute machine coding. Instructions and record areas are assigned actual memory locations. Constants are properly located for storage as program data. Information also may be taken from the library, if called for, in the form of subroutines or macro-instructions and included with the output program.

The autocoder assembly progresses through two phases of operation to produce the output program without intervention by the console operator. Assembly will continue without interruption under all normal operating conditions. Errors or other inconsistencies are noted on the typewriter.

The completely assembled output program may be written on tape and punched in cards, or it may be written on tape only. Loading instructions are produced in front of the output program to provide a convenient method of loading the program when it is placed into the 705 for actual operation. Figure 5 is a sample output program card record containing the identification of the program, the serial number of the card, the number of columns of program information punched, and the actual memory location where that information is to be stored. Up to 13 instructions in machine code or 65 characters of other program data may be contained in a single program record. Identification, columns 75-80 of the autocoder entry cards, is taken from the first entry processed.

Figure 6 is a sample of an output or subject program listing. All the information which was written on the original program sheet is listed for reference. In addition, the actual memory locations of instructions and data, machine operation codes, and instruction addresses are listed as produced by the assembly.

The printed list and the program cards are optional output from the assembly. A listing tape and a program tape are always produced.

When assembly has been completed, the output program can be changed, revised, or rearranged by a reassembly process. Either the output program listing tape or the original program tape is then used as input to the autocoder. Revision entries are punched into instruction cards and are fed into the reassembly in the same page and line number sequence as the original output program. Reassembly is also under control of the autocoder systems tape. The revised program will be produced as reassembly output in the same form as normal assembly output.

PROGRAM FORM

THE 705 autocoder requires two basic types of information to produce a finished program:

1. *Area Definition.* Reservation of memory space for input, output, and working areas which will contain the data to be processed or the various fixed factors or combinations of characters (constants) needed in executing the program.

2. *Instructions.* Commands to the 705 to operate upon data, constants, or other instructions using available components in the machine system.

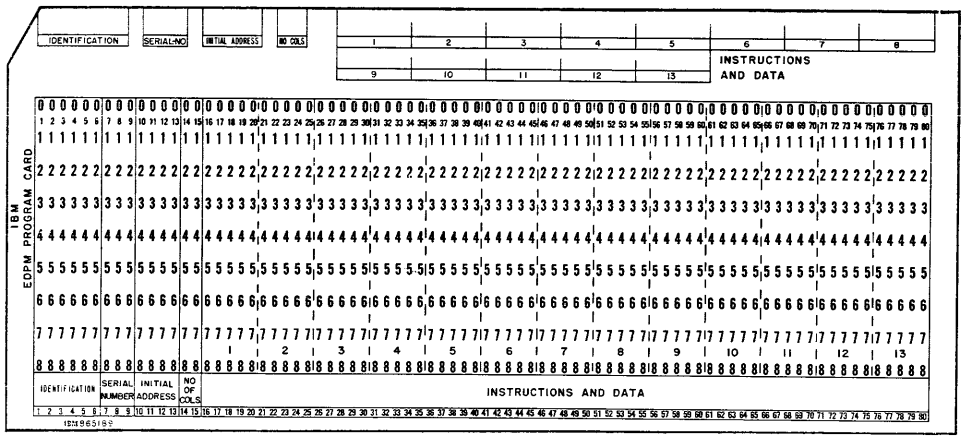

FIGURE 5. OUTPUT PROGRAM CARD

All program information is written on the autocoder program sheet, Form 22-6705-0 (Figure 1). The following sections explain the use of each column on the sheet.

Heading Line

A line is provided at the top of the page to identify and date the written program. The IDENT entry on this line is punched into columns 75-80 of all instruction cards in the program (Figure 2) which are input to the autocoder assembly.

Page Number

A two-character page number entry sequences the program sheets. Pages may be numbered alphabetically as well as numerically. The collating sequence of the 705 determines the order of the pages.

Line Number

A three-character line number sequences the program entries on each sheet. The first 25 lines are prenumbered from 010 to 250 to reduce the amount of written information required. Line numbering may continue beyond 250 for one sheet number up to the three-character limit of the field. The units position of the line number may be used for inserts. Blank lines are reserved at the bottom of the program sheet for this purpose.

The page and line number fields control the sequence in which the written program, as punched in cards or written on tape, will be fed into the autocoder assembly. Any 705 characters may be used in assigning page and line numbers. However, if alphabetic, numerical, and special characters are used, program entries should be in 705 collating sequence. Any variation from this sequence will be noted on the typewriter during assembly.

Tag

A program must be stored in memory when it is to be operated upon by the 705. The positioning of the program within available memory capacity is subject to certain restrictions which are inherent in the 705 itself. For example:

1. The units position of an instruction must always be placed in a memory location whose address ends in the digit 4 or 9.
2. Areas to be moved by five-character transmission must be evenly divisible by five.
3. Sufficient space must be reserved for records and other program data.
4. Some instructions are addressed to the right position of a field, others to the left position.

The tag field in the autocoder system is used to *represent* the location of program data in memory. Consequently, the programmer need not specify nor be concerned with actual memory locations. Information can be identified entirely by its particular label or tag. However, only items which are to be addressed by program instructions need to be tagged.

Each item to be referred to is assigned a distinctive and unique tag. These items may be records, areas, fields, instructions, or individual characters. Any item may be addressed after it has been assigned a tag.

During phase I of assembly, the autocoder assigns actual memory locations to the program data. Five memory positions are automatically assigned for each

```
PG LINE   TAG         OP     NUM    OPERAND          PROGRAM PAYRL1 COMMENTS          LOC OP ASU ADDRESS
01 010 GROSSTONET  SEL          202              READ INPUT                00164 2    00202 0202
01 020             RD           PAYRECORD                                  00169 Y    00380 0380
01 030             TRS          ENDOFJOB                                   00174 O
01 040             TRA          READERROR                                  00179 I
01 050             RAD    1     #&0000#          SET & FILL ASUS           00184 H  1 00423 04S3
01 060             RAD    2     #&000#                                     00189 H  2 00426 04K6
01 070             RAD          TAXCLASS          COMPUTE EXEMPTION AMOUNT 00194 H    00385 0385
01 080             MPY          #-1300#                                    00199 V    00430 0430
01 090             ADD          GROSSPAY                                   00204 G    00406 0406
01 100             TRZ          GTN1              TEST FOR WITHHOLDING TAX 00209 N    00219 0219
01 110             TRP          GTN2                                       00214 M    00229 0229
01 120 GTN1        ST     1     W/HTAX            NO WITHHOLDING - STORE ZEROS 00219 F  1 00410 04/0
01 130             TR           GTN3                                       00224 1    00254 0254

01 140 GTN2        MPY          #&18#             COMPUTE WITHHOLDING TAX  00229 V    00432 0432
01 150             RND          2                                          00234 E    00002 0002
01 160             SET          4                                          00239 B    00004 0004
01 170             ST           W/HTAX                                     00244 F    00410 0410
01 180             ADM          YTDW/H                                     00249 6    00397 0397
01 190 GTN3        RAD          #&420000#         TEST FOR FICA TAX        00254 H    00438 0438
01 200             SUB          YTDGROSS                                   00259 P    00391 0391
01 210             TRZ          GTN4                                       00264 N    00274 0274
01 220             TRP          GTN5                                       00269 M    00284 0284
01 230 GTN4        ST     2     FICA              NO FICA - STORE ZEROS    00274 F  2 00413 04J3
01 240             TR           GTN6                                       00279 1    00334 0334

01 250 GTN5        SUB          GROSSPAY                                   00284 P    00406 0406
02 010             TRP          GTN7                                       00289 M    00304 0304
02 020             ADD          GROSSPAY          COMPUTE PARTIAL FICA     00294 G    00406 0406
02 030             TR           GTN8                                       00299 1    00309 0309

02 040 GTN7        RAD          GROSSPAY          COMPUTE WHOLE FICA       00304 H    00406 0406
02 050 GTN8        MPY          #&2#                                       00309 V    00439 0439
02 060             RND          2                                          00314 E    00002 0002
02 070             SET          3                                          00319 B    00003 0003
02 080             ST           FICA                                       00324 F    00413 0413
02 090             ADM          QTDFICA                                    00329 6    00401 0401
02 100 GTN6        RAD          GROSSPAY          COMPUTE NET PAY          00334 H    00406 0406
02 110             ADM          YTDGROSS                                   00339 6    00391 0391
02 120             SUB          W/HTAX                                     00344 P    00410 0410
02 130             SUB          FICA                                       00349 P    00413 0413
02 140             ST           NETPAY                                     00354 F    00418 0418
02 150             SEL          203               WRITE RESULT             00359 2    00203 0203
02 160             WR           PAYRECORD                                  00364 R    00380 0380
02 170             TRS          SWITCHTAPE                                 00369 O
02 180             TRA          WRITEERROR                                 00374 I
02 190             TR           GROSSTONET                                 00379 1    00164 0164

02 200 PAYRECORD   DRCD                           DEFINE PAY RECORD        00380
02 210 MANNO                    5    &                                     00384
02 220 TAXCLASS                 1    &                                     00385
02 230 YTDGROSS                 6    &                                     00391
02 240 YTDW/H                   6    &                                     00397
02 250 QTDFICA                  4    &                                     00401
02 251 GROSSPAY                 5    &                                     00406
02 252 W/HTAX                   4    &                                     00410
02 253 FICA                     3    &                                     00413
02 254 NETPAY                   5    &                                     00418

       SIGNED LITERAL           1                                          00419
       SIGNED LITERAL           4    000&                                  00423
       SIGNED LITERAL           3    00&                                   00426
       SIGNED LITERAL           4    130-                                  00430
       SIGNED LITERAL           2    1H                                    00432
       SIGNED LITERAL           6    42000&                                00438
       SIGNED LITERAL           1    B                                     00439
```

FIGURE 6. OUTPUT PROGRAM LISTING

instruction and additional space is reserved for other areas according to their size. The programmer merely specifies the length of all such miscellaneous data.

Tags may be made up of ten characters or less in any pattern of numbers, letters, and special characters acceptable to the 705, including blanks, with the following exceptions:

1. The first character may not be a # (used as a parenthesis in instruction card punching), ⌑ (lozenge), or @ (commercial "at").
2. An & (ampersand) or a - (hyphen) may not be used in the tag field.
3. The tag may not be ten 9's.
4. Tags of library subroutines are made up of a

five-character name of the subroutine followed by up to five characters. The name portion of the subroutine tag should not be used as the first five characters of any other tag in the program.

Operation

The 705 can perform 35 different operations involving data which either are stored in memory or are handled by the various components of the machine. Each of these operations is assigned a single-character code. When stored in memory, the code can be interpreted by the machine as an order to calculate, read, write, compare, and so on. A complete machine instruction is always made up of two parts:

1. The single-character operation code.
2. The four-character address.

In the autocoder system it is unnecessary for the programmer to use any actual machine codes. Each operation has been given a mnemonic abbreviation which suggests the machine operation to the programmer. For example, the operation set left is abbreviated as SET, the operation load as LOD, and so on. All 705 operations have been abbreviated to either two- or three-character mnemonics. The autocoder will accept either machine codes or mnemonics, which ever are written into the program.

A number of operations are also available from the autocoder itself. These include special operations which assign memory areas to records, fields, and constants, produce address constants, use macro-instructions, call for subroutines in the library, or do other operations necessary to produce a finished program.

Figure 7 is a list of all 705, TRC, and high-speed printer operations with corresponding autocoder abbreviations. It gives a brief description of each operation together with the corresponding actual 705 single-character code.

Figure 8 lists all autocoder operations which are available, including those which call for subroutines from the library or are used to revise the program during reassembly or during revision of the library. Macro-instruction operations presently available are listed in the section "Macro-Instructions." As additional instructions are added to the library, the corresponding operations are also available.

Numerical

The numerical column of the program sheet:

1. Adjusts the memory location to which the first character of a defined area will be assigned. A zero in the numerical column, for example, properly locates an area for five-character transmission.

2. Specifies the accumulator or an ASU with operations such as RAD, SET, and so on.

3. Specifies the length of a record, record field, constant or other area.

4. May be used to sign the units position of an instruction.

The numerical entry, in combination with the operand, forms the address portion of an instruction.

Operand

The operand field is used in any of the following ways:

1. As an actual memory location, thereby providing the programmer with the ability to assign all, or portions of, the program or data to some predetermined specific section of memory. It may also be the actual address of data or of a machine component, such as a tape unit, drum section, check indicator, or alteration switch.

2. As the descriptive location of data or other instructions in memory. In this case, the operand is the exact duplicate of the tag of the data or instruction to which it refers. If there is no tag which exactly corresponds to a descriptive operand, the autocoder may use a tag which is very similar to the operand and which differs from it only in the position of a blank. A message will notify the programmer that an exact duplicate could not be found and that a substitution was made. The descriptive operand may also be the name of a machine component, provided that the component has been assigned this name by a special TRANS entry.

3. As the data to be operated upon. This form of operand relieves the programmer of any planning for memory reservation of constant factors. The autocoder, for example, permits the programmer to write an instruction with a literal operand as RAD(+1), without regard to where +1 will be stored. The assembly assigns the factor +1 to a memory location and automatically substitutes the address of this loca-

Instruction	Mnemonic Code	Oper. Part	Description
			705 OPERATIONS
Add	ADD	G	Result in accumulator or aux. stor.
Add Mem	ADM	6	Signed memory field--algebraic addition Unsigned mem. field--non-algebraic and zone addition
Compare	CMP	4	Accumulator or aux. stor. with memory
Ctrl 0000	IOF	3	Turn off input-output indicator
Ctrl 0001	WTM	3	Record tape mark
Ctrl 0002	RWD	3	Rewind tape unit
Ctrl 0003	ION	3	Turn on input-output indicator
Ctrl 0004	BSP	3	Back space tape one record
Ctrl 0005	SUP	3	Prevent printing or punching one cycle
Divide	DIV	W	Dividend in acc. 00; divisor in mem.; quotient in acc. 00
Lengthen	LNG	D	Add zeros to right (accum. 00)
Load	LOD	8	Loads zones as well as numbers
Multiply	MPY	V	Multiplier acc. 00; multiplicand mem.; product acc. 00
No Oper	NOP	A	No operation
Norm Tr	NTR	X	Left-hand zero in accum. or aux. stor. causes transfer. Zero deleted except last zero
Read 00	RD	Y	From tape, card reader, or drum to memory
Read 01	RD	Y	Pass tape record without storing in memory
Read-Write	RWW	S	Alert reading tape unit to operate simultaneously with next write instruction
Receive	RCV	U	Use with TMT for mem. to mem. transfer (See TMT)
R Add	RAD	H	Resets accum. or aux. stor.; adds from memory
R Sub	RSU	Q	Resets accum. or aux. stor.; subtracts from memory
Round	RND	E	Drop positions from right and 1/2 adjust (accum. 00). Zero balance is plus
Select	SEL	2	Select input, output, alter sw. or check indicator
Set Left	SET	B	Place storage mark to left of indicated address (00-15)
Shorten	SHR	C	Drop positions from right (accum. 00). Zero balance is plus
Sign	SGN	T	Remove zone 11, 10, or 01 from memory, place & or -- in accumulator or aux. stor.
Stop	HLT	J	Stop machine
Store	ST	F	Store numerical field in memory
St Print	SPR	5	Store field for printing. Skips decimals, commas. Removes left-hand zeros and commas
Subtract	SUB	P	Result in accumulator or aux. storage
Transfer	TR	1	Unconditional transfer
Tr Any	TRA	I	Transfer if any I/O or check indicator is on
Tr Equal	TRE	L	Transfer if acc. or aux. stor. is equal to memory
Tr High	TRH	K	Transfer if acc. or aux. stor. is higher than memory
Tr Plus	TRP	M	Transfer if acc. or aux. stor. sign is plus
Tr Signal	TRS	O	Transfer if indicator is on
Tr Zero	TRZ	N	Transfer if factor in acc. or aux. stor. is zero
Transmit	TMT	9	Use with RCV for transmitting data from memory to memory, 5-character group or single character
Unload	UNL	7	Unload zones and numbers
Write 00	WR	R	From memory to tape, card punch, printer, or drum
Write 01	WR	R	Dump memory to card, printer, tape, drum
Wr Erase	WRE	Z	Replace each char. in mem. with blank and write on tape, card punch, printer, or drum
			760 OPERATIONS
Ctrl 0026	RWS	3	Read/Write start
Ctrl 0027	RWT	3	Read/Write tape
Ctrl 0028	RST	3	Reset counter
Ctrl 0029	PTW	3	Print tape write
			777 OPERATIONS
Ctrl 0015	PRW	3	Prepare to read while writing
Ctrl 0016	RTS	3	Read tape to TRC
Ctrl 0017	WST	3	Write TRC to tape
Ctrl 0018	BPC	3	Bypass TRC

NOTE: Control instructions may also be written as CTL with proper address.

FIGURE 7. TABLE OF MACHINE OPERATIONS

tion for the literal operand. The operand may be a signed or unsigned field. The autocoder will define all signed literals as arithmetic fields, separate them from unsigned literals, and enter all literals in a table at the end of the program data. A literal will appear in the table only once even though it may appear several times in the program.

AUTOCODER OPERATIONS	
OPERATION	DESCRIPTION
DRCD	Define record
DCON	Define constant
DFPN	Define floating point number
AACON	Actual address constant
LACON	Left address constant
RACON	Right address constant
TITLE	Title entry
TRANS	Translate tag to address
LDDR	Load program on drum
LDTP	Load program on tape
LLDTP	Load program on tape, WTM, RWD
LASN	Location assignment
SASN	Special Assignment
DELET	Delete from library or delete during reassembly
INSER	Insert in library
REPLA	Replace library
INCL	Include sub-routine
DO	Include sub-routine and transfer unconditionally
DOA	Include sub-routine and transfer on any
DOE	Include sub-routine and transfer on equal
DOH	Include sub-routine and transfer on high
DOS	Include sub-routine and transfer on signal

FIGURE 8. TABLE OF AUTOCODER OPERATIONS

Operands can be written which refer to any character position within an instruction, field, or record by using character adjustment. The adjustment is a plus or minus factor written after the operand. An address may be adjusted in such a manner that every position of memory may be referred to by the program.

Comments

A comments field is provided where additional information concerning the program may be included if desired. Ordinarily comments are transcribed to the output program listing for reference only.

The comments column may also contain constants if special constant areas have been defined. The operands of certain macro-instructions may overflow into the comments column depending upon the number of parameters which must be supplied to a particular instruction.

AUTOCODER PROGRAMMING

THIS section describes in detail the various steps to be followed for the preparation of the written autocoder program. Methods of assigning or "defining" records, fields, constants, and work areas are explained, together with the rules to be followed in writing 705 instructions in autocoder language.

AREA DEFINITION

MEMORY areas should be reserved for the storage of all records and any other data which are to be processed by the program. Such records and data are normally made up of various fields which are of known length and arrangement. Records will be automatically assigned to memory locations in the order in which they are defined in the program. However, they can overlap other areas if so specified by the programmer. Program instructions which perform operations upon records are addressed to the specific memory positions where these records are located.

A simplified payroll record input and output area is illustrated in Figure 9. The arrangement of fields and the number of positions within each field is shown. A total of 35 positions of memory will be needed to store the input record, and 44 positions to store the output record.

Record layout description is written on the program sheet as shown in Figure 10. The record name and the name of each field is written in the tag column and the number of positions in each field (length) in the numerical column.

Tag

The record may be given a tag to provide a reference for instructions, such as read and write, which address the entire area. It is usually an advantage to choose tags which are descriptive of the area to which they are assigned, e.g., PAYROLL IN, WITHDRAWALS, RECEIPTS, and so on. Such tags not only provide an easily remembered reference for the programmer, but also assist others who may be called upon to follow the program for testing, correction and actual operation.

Each individual field is also given a tag if that field is to be addressed by program instructions. These tags provide program reference for instructions which are normally concerned with individual fields in the record, such as compare, load, arithmetic instructions, and so on. Fields which are common to several different areas may be given distinctive tags by using an alphabetic or numerical prefix or suffix, as written for input and output areas in Figure 10.

As a general rule the tag column should not be used only for description. Such descriptive information should be placed in the comments column.

Operation

The operations DRCD (define record), DCON (define constant), and DFPN (define floating point number) initiate the memory assignment of individual memory areas during assembly. The arrangement of fields

PAYROLL RECORD - INPUT A

FIELDS	Pay Per	Dept. No.	MAN NO.	Tax Cl.	Y.T.D. EARN.	Y.T.D. WITH.	QUAR. FICA	GROSS PAY	bbb‡
	x x	x x	x x x x x	x	x x x x x x	x x x x x x	x x x x	x x x x x	x x x x
LENGTH	2	2	5	1	6	6	4	5	4

PAYROLL RECORD - OUTPUT B

FIELDS	Pay Per	Dept. No.	MAN NO.	Tax Cl.	Y.T.D. EARN.	Y.T.D. WITH.	QUAR. FICA	GROSS PAY	CUR. W. H.	CUR. FICA	NET PAY	‡
	x x	x x	x x x x x	x	x x x x x x	x x x x x x	x x x x	x x x x x	x x x x	x x x	x x x x x	x
LENGTH	2	2	5	1	6	6	4	5	4	3	5	1

FIGURE 9. PAYROLL RECORDS, PARTIAL LAYOUT

within the record is described on the program sheet by entries written on immediately following lines (Figure 10).

DRCD, DCON and DFPN entries are prepared in the same manner. The following conditions apply.

A tag may or may not be supplied, depending upon whether reference is made in the program to the entire area or only to individual fields within the area. The actual memory address which will be calculated by the autocoder for the entire defined area is that of the last character of that area, which becomes the address of the area for all low-order reference instructions, such as LOD. Addressing the entire area by a high-order reference instruction, such as read or write, however, results in the address of the first character of the defined area.

For example, the payroll record input area in Figure 10 may be addressed by a read instruction by writing the instruction RD PAY REC IN. RD is the conventional 705 operation code; PAY REC IN is the address of the input record.

After the location of the last character of an area has been established, it is then used to calculate addresses for all instructions referring to the entire area. For example, if the last character of PAY REC IN is assigned to location 4084, then the address of a load instruction is 4084. The address of a read instruction, however, becomes 4084 (location of last character) —35 (length of all the fields in the record) +1, or 4050. The look-up and calculation of addresses is accomplished during phase II of the assembly.

The operation column may contain DRCD, DCON, or DFPN, depending upon the type of area to be defined.

Entries describing record fields, constants and floating point numbers are written on lines which follow the DRCD, DCON, DFPN entries. The operation column of an entry following a DRCD, DCON, or DFPN is blank except when the length of the entry is greater than 99.

The entire operation column remains blank when describing floating point numbers.

Numerical

The numerical column in the DRCD, DCON, and DFPN entries specifies the proper memory location to which the first character of the defined area will be assigned. If blank, the area will start at the current setting of the assignment counter. If a digit from 1 to 4 is used, the first character will be located at the next memory address ending in 1 or 4, or at an address ending in 1 to 4 plus 5, whichever is more conserva-

LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
010	PAY REC IN	DRCD	0		Payroll record input
020	APAY PER		2		
030	ADEPT NO		2		
040	AMAN NO		5		
050	ATAX CL		1	+	
060	AYTD EARN		6	+	
070	AYTD WITH		6	+	
080	AQUAR FICA		4	+	
090	AGROSS PAY		5	+	
100	AREC MARK		4		
110	PAY RC OUT	DRCD	0		Payroll record output
120	BPAY PER		2		
130	BDEPT NO		2		
140	BMAN NO		5		
150	BTAX CL		1	+	
160	BYTD EARN		6	+	
170	BYTD WITH		6	+	
180	BQUAR FICA		4	+	
190	BGROSS PAY		5	+	
200	BCUR WH		4	+	
210	BCUR FICA		3	+	
220	BNET PAY		5	+	
230	BGR MARK		1		
240					

FIGURE 10. PAYROLL RECORDS, DEFINITION

tive of memory. If the digit 0 is used, the first character will be properly located for high speed transmission at an address ending in 0 or 5, whichever is more conservative of memory. The numerical column should be left blank when a descriptive, or actual address is used in the operand field.

The length of each record field or constant is written in the numerical column of the entries following a DRCD, DCON, or DFPN entry. Memory space is allocated equal to the number of positions specified for each field. When a field length exceeds 99 positions, overflow from the two-digit numerical column on the program sheet may extend up to three places into the operation column. The total space in memory assigned to a defined area is equal to the total lengths of the various fields which make up that area. Field lengths may be specified up to the capacity of memory.

The length of a constant field is equal to the number of characters in the constant. When a length is specified which is less than the actual number of characters in the constant, the autocoder will shorten the constant to the specified length by omitting right-hand characters. When a length is specified which is greater than the actual constant, the autocoder will add zeros to the right of a signed constant, and blanks to the right of an unsigned constant. Plus or minus signs preceding the constant are not counted as characters when specifying length.

The numerical column remains blank for floating point numbers. The assembly supplies a length of 10.

Operand

The operand column of the DRCD, DCON or DFPN entry specifies the location of the defined area in memory. If the field is left blank, the record area will immediately follow the location of preceding entries, and this area will be reserved to prohibit other entries from being placed there.

However, the programmer may, if he wishes, instruct the autocoder to assign areas to some specified location in memory. In this case it will be the responsibility of the programmer to make sure that nothing else is improperly placed in that location.

A record area may be located in any specified portion of memory by placing the actual address in the operand column. Actual addresses are preceded by the symbol @ on the program sheet. The first character of the record will be located in the actual memory address specified.

A previously assigned tag may be used as an operand. In this case the area will be assigned the same locations as the entry identified by that tag.

A defined area can be assigned to a location which follows in line with the regular sequence of instructions and other areas; it can be assigned to a specific prelocated portion of memory, or it can overlap some previously defined area. In the latter case, the longer area should be defined first to avoid overlapping beyond the proper portion of memory.

The use of the operand column is optional when describing fields within records. Information written will be key punched into the instruction cards and will be printed on the output program listing for reference.

A plus or minus sign should be written as a one position operand in a field within a DRCD area that is to be addressed by an arithmetic operation. During assembly the autocoder checks to see that the field which is addressed by arithmetic operations (exclusive of LOD, UNL, ST, and ADM) has been written with a sign as the first position of the operand. If this is not done, a sign check message will be typed during assembly. This check aids the programmer in manually checking that all such work areas are signed by the program or that the data are signed.

To set up constants in memory, write DCON in the operation column. Starting with the next line. write the constants in the operand column, one constant to a line. Any 705 characters are acceptable as constants. The constant may extend into the comments column and up to 52 characters may be written (including the sign) as one entry. Constants longer than 52 characters must be written as two or more entries with the proper length specified for each.

A numerical constant field may be signed by writing a plus or a minus sign *preceding* the field on the program sheet. Only numerical constants should be signed. This results in the signing of the last digit of the field. Plus signs are punched as & and minus signs as — in the instruction cards and are properly interpreted by the autocoder assembly.

Figure 11 is the complete record and constant layout which might be used to process the payroll records shown in Figure 9. A print area and a work area have been included to permit total accumulation, by department, of gross, tax, and net pay. A constant

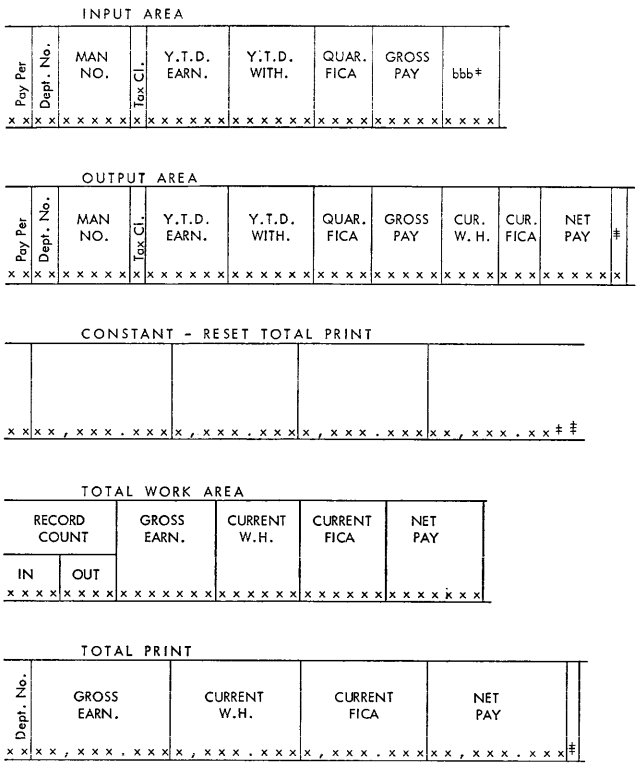

FIGURE 11. PAYROLL RECORDS, COMPLETE LAYOUT

print pattern is placed in memory to reset total print areas. Figure 12 shows how the constant pattern may be written on the program sheet. The figure also shows how the constant + 4200 would be set up.

One or more floating-point-number entries may appear following the operation DFPN. Each floating point number is written in the form

$$\pm \text{ e e } \pm \text{ xxx . . .}$$

in which *e* stands for the exponent on the base 10 and xxx. . . is a mantissa between 0 and .99999999. For example, the number 365 would be written as + 03 + 365. Up to eight mantissa places may be specified. However, if fewer than eight are entered, the autocoder will complete the number by supplying zeros. The representation of the quantity zero is written as — 99 + 0.

It should be remembered that DRCD entries differ from DCON and DFPN entries in that only the latter produce output program cards.

Comments

The use of the comments column is optional with record and field definition entries. The column can be used for recording program notes or other information which may clarify the logic or function of the program. The tag may serve as an adequate label of the various areas involved but should not be used as a substitute for comments.

Entries in the comments column are punched into the instruction cards and will be printed on the output program listing exactly as originally written by the programmer.

Constants may extend into the comments columns if necessary.

INSTRUCTIONS

ALL instructions which can be performed by the 705 are valid input to the autocoder assembly. Instructions are written on the program sheet, one instruction per line, in the exact sequence in which they are to be executed by the machine. The correct sequence of input is assured by sorting all instruction cards to page and line number before assembly.

Tag

The assignment of instruction tags is subject to the same rules previously outlined for the assignment of tags to record and constant areas. Only instructions which are addressed by other instructions in the program need to be tagged.

It is usually most convenient to form tags which are descriptive of the section of the program which is to be addressed. For example, the first instruction of a routine to calculate net pay can be tagged NET CALC. The tag then becames an easily remembered descriptive reference point. A transfer to this routine can then be written as TR NET CALC. Switches, digit selectors, and instructions which are to be modified by other instructions may be tagged with the actual name of the reference point, e.g., SW 1, DIG SEL 1, ALPHA LOD, and so on. The autocoder assembly con-

LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
010		DCON			
020	PRINT		41	bbbb,bbb.bbbb,bbb.bbbb,bbb.bbbbb,bbb.bb‡‡	
030	CONSTANT 1		04	+4200	

FIGURE 12. DCON ENTRY, PRINT AREA

verts all tags to actual memory locations during phase I. Operands which refer to these tags are converted to addresses during phase II.

It is often convenient to relate program instructions directly to the corresponding blocks of a flow diagram. This procedure furnishes cross reference between the diagram and the program and is an aid to program testing and correction. Reference between the program and the diagram may be provided by the following procedure:

1. Repeat the page and line number as a tag. This method furnishes a sequenced reference which may be easily keyed to the flow diagram. In this case, however, the advantage of the descriptive tag is lost.

2. Use descriptive tags carrying their own sequence indication, such as NET CALC 1, NET CALC 2, NET CALC 3, and so on. This method combines the advantages of both the sequence and descriptive features.

3. Insert title entries at various points in the program. These entries can also be referenced to the flow diagram where convenient. (Refer to "Title" section.)

However, as a general rule, the tag column should not be used for description only. Such descriptive information should be placed in the comments column.

Operation

All 705 instructions with corresponding mnemonic abbreviations are listed in Figure 7 and are described in detail in the 705 *Preliminary Manual of Operation*, and in the manuals of operation on the 777 and 760. Either the mnemonic abbreviation or the single-character code is acceptable as autocoder input.

The autocoder processes all 705 instructions according to the kinds of operands with which each may be associated. Figure 13 lists the six groups of instructions together with their permitted types of operands. The proper character position addressed in memory is also shown.

Autocoder operations may also be written into the operation column of the instruction. Some of the operations, such as the assignment of areas to specific sections of memory, place certain functions of the assembly under the control of the programmer. These are explained in the section "Special Operations." Other autocoder operations call for subroutines from the library. Up to five characters may be used.

An operation may also be a macro-instruction name, causing the assembly to substitute a sequence of conventional 705 instructions from the library.

Group	705 Instructions	Permitted Operands	Character Addressed
1	ADD RAD ADM RSU CMP SGN DIV SPR LOD ST MPY SUB NOP UNL	Blank Actual (5-digit max.) Literal Descriptive	Low-order
2	RD RWW WR WRE RCV 01-15 TMT 01-15	Blank Actual (5-digit max.) Literal Descriptive	High-order
3	RCV 00 TMT 00	Blank Actual (5-digit max.) Descriptive	High-order +4
4	NTR TRH TR TRP TRA TRS TRE TRZ	Blank Actual (5-digit max.) Descriptive	Low-order
5	CTL SEL HLT SET LNG SHR RND	Blank Actual (4-digit max.) Literal Descriptive	
6	BPC RTS BSP RWD IOF RWS ION RWT PRW SUP RST WTM	Blank. The operation code 3 and the proper numerical address are supplied by the Autocoder.	

FIGURE 13. INSTRUCTIONS BY AUTOCODER CLASS

Numerical

The numerical column is used to specify an ASU or the accumulator. Either one or two digits, from 1 to 15, may be written. A blank numerical column, 0, or 00 will denote the accumulator. An ASU designation may be written as 4 or 04.

The numerical column must be considered when using the RCV instruction. Unlike actual 705 usage, the autocoder requires ASU designation with the RCV instruction to determine whether five-character or serial transmission is involved and to calculate the address accordingly. The same ASU should be specified for both corresponding RCV and TMT instructions.

Operand

The instruction operand is written in the operand column of the program sheet. It is converted to an address in phase II of the autocoder assembly. Instruction operands may be of several types.

ACTUAL

Actual operands may designate an actual memory address. In this case, the operand must be preceded

by an @ symbol. Any address may be specified from 0 to the limit of memory. Insignificant zeros need not be written for actual addresses; the assembly supplies necessary zoning if a five-digit address is specified. For example, location 00250 is written as actual operand @250; location 25,000 as @25000.

Instructions such as SET or SEL (group 5, Figure 13) which normally have actual operands, may, but do not have to, be preceded by the @ symbol.

LITERAL

A literal operand is literally the data which are to be operated upon by the instruction. It may consist of from one to fifty characters, not counting a sign.

The use of literal operands eliminates the need for preliminary layout of constant areas, messages, and miscellaneous factors before writing the program. Such fields may be entered as literal operands when needed.

For example, assume that a factor +1300 is needed for calculation of a withholding tax exemption amount. To introduce this factor into the program, write the factor as (+ 1300) in the operand column of the instruction which uses the factor (Figure 14), enclosing it in the parentheses to indicate that it is a literal. Parentheses are key punched as # signs and will appear as # signs on the final program listing. The first character of the literal is its sign. During assembly, the factor +1300 is properly assigned a memory location as a signed numerical field. The location of the field becomes the address of the instruction. Figure 14 illustrates the proper method of writing literal operands.

Note that the instruction to write an end-of-file message includes the group mark as the last character of the message. The literal operand of the unload instruction reserves a 13-character field in memory where a control or indicative field might be stored for comparison with other fields from following records. Any field may be similarly reserved by writing an operand containing a number of characters equal to the number of memory positions needed for storage.

There are no restrictions on the use of valid 705 characters within literals. The literal is called "signed" if its first character is a plus or minus sign. The same literal may be used any number of times within the same program. However, the literal will be assigned only one location by the autocoder assembly and will appear only once in the final output program arrangement. The signed literals are grouped together at the end of the program. A blank precedes the group; the unsigned literals follow the signed literals. Literals cannot be located for high-speed transmisison. There is a limit to the number of different literals that can be assembled by the autocoder. If this limit is exceeded, succeeding literals will be ignored. If the average length of the literals used is five, about 250 distinct literals can be used. When it appears that this limit may be exceeded, it is suggested that the longer literals be set up as DCON entries.

Because literals are enclosed within parentheses (translated as # by autocoder), their lengths are limited to 50 positions, or 49 characters plus a sign, which is the limit of the columns in the instruction card.

NOTE: Working storage literals which may be *modified during the actual running of the program* should always be entered as a field with at least one non-numerical character, or they should be preceded by a blank if a pure numerical field is used. This convention avoids the possibility of conflict if a subroutine or macro-instruction should generate a literal that is a duplicate of one used in the main routine.

DESCRIPTIVE

The descriptive operand corresponds to a tag that appears elsewhere in the same program. The operand may be the tag of any area, field, or instruction.

LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
010	WH TAX	RAD	1	(+1300)	Get exem factor
020					
030		WR		(END OF FILE‡)	EOF message
040					
050		UNL		(COMPARE FIELD)	
060					
070		ADM	4	(+0005)	Address modify factor
080					

FIGURE 14. LITERAL OPERAND ENTRIES

Restrictions on the use of certain characters in tags also apply to descriptive operands. Figure 15 illustrates the use of descriptive operands.

BLANK

The blank operand always has blanks in the first ten positions of the operand field. Such instructions are to have proper addresses inserted by the running program. No ASU zoning may be used with blank operands and character adjustment does not apply.

OTHER

When a descriptive operand is used, reference is usually made to the location in memory of the data designated by the descriptive operand. However, the operands of instructions such as CTL, HLT, LNG, RND, SEL, SET, and SHR refer to machine components or functions, the length of accumulator fields, and so on. This class of instructions may carry descriptive operands, however, if the operand is the tag of some field defined in the program.

For example, in a program where the tag MANNO has been defined to be of length 6, the sequence of autocoder instructions:

```
SET   15   MANNO
LOD   15   MANNO
```

will cause ASU 15 to be set to 6 and the field MANNO loaded into ASU 15.

Machine components may also be addressed with descriptive operands, provided that such components have been defined by a TRANS operation. Refer to section "Special Operations."

Normally when a literal operand is used, the literal is the actual data to be acted upon by the 705 instruction. However, instructions such as CTL, HLT, LNG, RND, SEL, and SHR may have literal operands. In such cases the actual data is not referred to, but reference is made to the number of positions of the literal.

For example, in the following program:

```
SET   (01298)
LOD   (01298)
```

the autocoder will translate this as:

```
SET   0005
LOD   Address assigned to 01298 by
        the autocoder.
```

For group five instructions only (CTL, HLT, and so on), the autocoder assumes an operand to be actual if it is wholly numerical and of a length less than five characters. An operand is considered descriptive if it contains a non-numerical character or is of a length greater than four characters. The autocoder recognizes only those characters to the left of the first blank in the operand field in making this determination. Therefore the operands bAB or 1bAB would be considered as actual addresses, and assembled as 0000 and 0001, respectively.

Character Adjustment

Any memory position within a designated field, area, or instruction may be addressed by the use of character-adjusted operands. This feature is useful, for example, in functions of address modification when it is desirable to modify only the operation code or the address portion of an instruction.

One common use of address modification is to change a program switch setting by changing the operation code from NOP to TR. This is done by changing the NOP code, character A, to the TR code, character 1, by removing the zoning from A with a sign instruction.

The instruction on line 010 in Figure 16 is a program switch. With its operation code set to NOP (character A), the instruction will not be executed.

LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
010		RD		PAY REC IN	Read in pay record
020		TRA		RD TRA	To error routine
030					
040		ADM		AQUAR FICA	Adjust quar FICA
050					
060		RCV		PAY RC OUT	Move pay record to
070		TMT		PAY REC IN	output area
080					
090		MPY		BTAX CL	Calc exem amt
100					

FIGURE 15. DESCRIPTIVE OPERAND ENTRIES

With its operation code set to TR (digit 1), a transfer will be made to the instruction tagged READ CARD.

A SGN instruction to set the switch to TR is shown on line 040. The operand is the tag SW 1 with a —4 character adjustment. The autocoder interprets the operand with the adjustment factor as the address of the SW 1 instruction minus four, or the address of its operation code. Lines 070 and 080 show instructions with character-adjusted operands to set SW 1 to NOP.

All descriptive operands may be character adjusted. The character adjustment factor is always written after the operand to which it applies. The number and direction of character adjustment is indicated by a plus or minus sign followed by the factor, which may be up to five digits in length.

To adjust an operand to four higher positions of memory, the adjustment factor is written +4. To adjust an operand four lower positions of memory, the adjustment factor is —4. Character adjustment proceeds from the high-order position of an area when used with operations such as RD, WR, RCV, TMT, and from the low-order position when used with operations such as RAD, LOD, CMP, and so on.

The sign of the character-adjustment may be located anywhere after the significant portion of the operand itself, but no farther to the right than the eleventh position of the *entire* operand field. An operand field may be written like any of the following:

PAY+4
PAYbbb+4
PAYbbbbbbb+4

Literal operands may also be character-adjusted, by placing the adjustment factor after the parentheses enclosing the literal. The sign of the adjustment may appear in any position after the parenthesis, up to, and including, the eleventh position of the *entire* operand field. Adjustments may be written for literals longer than ten positions if the adjustment factor immediately follows the literal. Adjustments to literal operands may be written as shown by instructions in Figure 17.

The instruction LOD on line 020 will result in loading the digits 1234 into ASU 10. The instruction on line 040 will write the phrase EOF UNIT 204 on the selected output unit.

NOTE: If the program is to be checked by EAM, it is advisable, wherever possible, to put all character adjustments in the 11th position of the operand field (viz., put the sign in column 33).

Comments

The comments field is normally reserved for comments which may be helpful in checking the program. The information in this field will have no effect upon the subject program produced by the autocoder. Exceptions are the following.

1. Certain macro-instructions may have several operands separated by a lozenge (⌑) character. These may extend into the comments field.
2. *ASUXX (any ASU 00-15) in the first six columns of the comments field of a 705 instruction has the special effect described under "Address Modification."

SPECIAL OPERATIONS

SPECIAL operations other than DCON, DRCD and DFPN are described in this section.

Title

The TITLE operation code permits the insertion of lines of descriptive information in the program. These lines are printed on the output listing for reference but do not in any way affect the operation of the program. They are not included on the output program tape or program cards.

LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
010	SW 1	NOP		READ CARD	To next transaction
020					
030					
040		SGN	1	SW 1-4	Set SW to TR
050					
060					
070		SGN	1	SW 1-4	Set SW to NOP
080		ADM	1	SW 1-4	
090					

FIGURE 16. ENTRIES FOR PROGRAM SWITCH SETTINGS

Any part of the tag, numerical, operand, and comments fields may be used for description. Title entries are useful as descriptive headings for various branches or sections of a program and may provide convenient reference points when the program is ready for testing and use. Figure 18 illustrates two title entries.

NOTE: Title entries may *not* be placed between an area and its field definition entries.

Translation

The operation TRANS may be used to establish direct program reference to a numerical operand field. An operand used with TRANS may be only an actual number from one to five digits.

For example, the first entry shown in Figure 19 is tagged MASTERTAPE with a TRANS operation. The operand in this case is the actual address of the master tape unit, 0204.

Any entry in the program which refers to the tag MASTERTAPE will cause the assembly to translate the tag into the given actual address. The second entry shown in Figure 19, SELECT MASTERTAPE, is therefore translated into SELECT 0204, in the output program.

Address Modification

The entry *ASUXX (any ASU 00-15) in the first six columns of the comments field of a 705 instruction will produce an ADM instruction as the next operation. The ADM operation will have xx in its numerical field and will operate on the preceding instruction.

The use of this feature for address modification presupposes prior steps in initializing the address to be modified. The address modifier must be in the specified ASU, and other programming must be written to control the number of executions.

Figure 20A shows an address modification entry which specifies ASU 04 in the comments column. The entries generated by the assembly are represented in Figure 20B.

Tape and Drum Loading

The following operations provide for the loading of the output program, or any selected part of the program, on a tape or drum:

LDDR	Load drum
LDTP	Load tape
LLDTP	Last load tape

The special load entry is always the last entry of that part of the program which is to be placed on the drum section or tape unit. The second operand of the load entry is the tag of the first entry to be loaded. The section loaded will therefore be defined as beginning with the entry specified by the load operand and ending with the load entry itself.

LDDR, LDTP, and LLDTP insert 00 control cards in the output program deck. (Refer to the section "Output Deck.") When the program is being loaded for actual operation, the loading is interrupted and instructions on the control cards are executed. These instructions place a group mark after the last character of the program just loaded, select the output unit (drum section or tape unit), write the blocks

LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
010		SET	10	4	Adjust ASU
020		LOD	10	(123456)-2	Get 1234
030					
040		WR		(FALSE EOF UNIT 204 ≠)+6	
050					

FIGURE 17. CHARACTER ADJUSTMENT TO LITERAL OPERANDS

LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
010		TITLE		GROSS TO NET PAYROLL	
020					
030					
040		TITLE		CALCULATE ORDERING QUANTITY	
050					

FIGURE 18. TITLE ENTRIES

LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
010	MASTERTAPE	TRANS		204	
020					
030					
040					
050					
060		SEL		MASTERTAPE	
070					

FIGURE 19. TRANSLATE ENTRY

LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
010	GET GROSS	RAD	15	GROSS PAY	*ASU04
020					
030				A	
040					
050					
060		RAD	15	GROSS PAY	
070		ADM	4	GET GROSS	
080					
090				B	

FIGURE 20. ADDRESS MODIFICATION

of the program on the selected unit, and then transfer back to the loading program. LLDTP also writes a tape mark and rewinds the tape. Any machine errors made in carrying out these operations will be detected by the load program.

If it is desired to load library subroutines on tapes or drum, it will be most convenient to include a LDTP, LDDR, or LLDTP as part of the subroutine. The literal table cannot be loaded on tapes or drum by LDDR, LDTP, or LLDTP.

The next portion of the program following the section loaded on drum or tape will be assigned to succeeding memory areas. If it is expedient to overlap this section, the load entry should be followed by an assignment entry whose operand is also the tag of the first entry of the drum or tape-loaded section. This operand may be character-adjusted to overlap the area partially if desired.

The instructions are written as shown in Figure 21, where:

1. X1 is the 4-digit tape unit or drum section address.
2. X2 is the address of the first character of the block of information to be written, either descriptive, actual, or blank (omitted).

If the address of the first character is omitted, the autocoder will assume that the entire program is to be loaded, starting at the first character of real information (a constant or instruction).

In all cases, the section to be loaded is assumed to end with the coding immediately preceding the tape or drum loading command. If the first character location specified is higher than the location just preceding the tape or drum loading command, it is an error and the results in the output program will be incorrect.

Assignments

The most important entries in the autocoder assembly are the LASN (location assignment) and SASN (special assignment). These are the only instructions whose misuse is likely to result in a wasted assembly. They are methods by which the programmer can control the assignment of 705 memory to instructions, constants, and records which are input to the autocoder.

LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
010	T_1	LDDR		X1 ⊔ X2 ⊔	
020					

FIGURE 21. LOAD DRUM ENTRY

The LASN entry is the signal to the autocoder to over-ride the automatic assignment of memory locations and to accept directions from the programmer where future entries are to be assigned. If an actual address is placed in the operand column, following entries will be placed starting at that actual location. For example, to begin the program assignments during assembly at memory address 1000, the first written program entry would be LASN@1000. If a tag is placed in the operand, the following entries will be placed overlapping that tagged item. The tag must have been defined previously in the program. If the operand of the LASN entry is left blank, the following entries will be placed immediately after the highest point already reserved in the assembly; or, to express this another way, the following entries will be placed in the first unreserved (blank) location in memory. The autocoder will at this point resume its automatic assignment process.

The LASN with a blank operand is normally used to terminate overlapping, or to terminate the effect of a SASN entry. Using the LASN to overlap, it is not necessary to write the largest area of programming first.

In normal autocoder processing a record is kept of the highest location to which any item has been assigned. This not only permits the LASN with a blank operand to function, but also will establish the beginning location of the subroutines (if any) and the literals. SASN permits the programmer to make assignments to any location in memory without affecting the high assignment counter.

SASN is normally used to place instructions in upper memory without forcing the assignment of subroutines and the literal table outside memory.

The overlap of program instructions must be planned by the programmer. When using an LASN entry which covers a preceding section of the program, the instructions which are thereby erased must be executed before they are covered. The autocoder does not automatically provide for interruption of the loading process so that a given routine may be worked before succeeding instructions are loaded into the same area. The programmer must either manually insert control cards into the program deck to control the loading, or he may save the covered instructions by loading them on a drum section or tape beforehand using LDTP, LLDTP, or LDDR.

For example, it is often desirable to overlap housekeeping instructions which may be executed only once during a job. The housekeeping instructions must be executed, however, before they can be covered by succeeding instructions. The loading process can be interrupted by control cards; housekeeping is then executed and loading continues.

Address Constants

The operation codes LACON, RACON, and AACON (address constant definitions), together with the proper operand, produce a five-character constant during assembly. The constant is made up of the character A followed by the four-character memory address of the location where the field referred to by the operand has been stored. An address constant will always begin at a 0 or 5 position and end at a 4 or 9 position.

The operation LACON produces the actual memory address of the left (high-order) character of the field referred to by the operand as an address constant.

The operation RACON produces the actual memory address of the right (low-order) character of the field referred to by the operand as an address constant.

The operation AACON operates like the code RACON with an actual operand. However, when using AACON, it is not necessary to insert the @ before the operand.

Address constants developed by LACON, RACON and AACON entries may have ASU zoning over the tens and hundreds positions or may be signed over the units digit. The ASU designation or the algebraic sign of the constant is entered in the numerical column of the program sheet. LACON, RACON, and AACON entries may be written within a sequence of instructions, because the character A of the constant functions as a no-operation. The address constant is, in effect, a no-operation instruction which is executed without affecting the program.

Three types of addresses may be used with address constants.

DESCRIPTIVE ADDRESS

This is the address of a record area, field, constant, or instruction. Figure 22A shows a RACON entry used to develop the address of the BNET PAY field shown in Figure 10. The entry is tagged ADDRESS 1 so that the constant may be addressed by other program instructions. The constant address is to be used as an instruction specifying ASU 14.

LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
010				A	
020	ADDRESS 1	RACON	14	BNETPAY	Descriptive constant
030				B	
040	ADDRESS 2	RACON	10	@24629	Actual constant
050	ADDRESS 3	AACON	10	24629	Actual constant
060	ADDRESS 4	RACON	+	@200	Actual constant
070				C	
080	ADDRESS 5	LACON		(1300)	

FIGURE 22. ADDRESS CONSTANT ENTRIES

The tag of a program instruction may also be used as a LACON or RACON operand. This device is useful in developing the constant addresses of program instructions. Such constants may be used to perform the various functions of address modification.

ACTUAL ADDRESSES

The operand of the LACON or RACON entry may be an actual 705 address. In this case, the first character written on the program sheet must be an @ followed by up to five digits of an actual address. ASU or sign designation may be placed in the numerical column. Figure 22B shows two entries to develop the actual address 24629 as a constant specifying ASU 10. An entry to develop the address of tape unit 200 as an algebraic constant is also shown.

NOTE: In this case the actual four-character constant, not its address, is placed in memory. The constant is preceded by the character A.

LITERAL ADDRESS

The operand of LACON or RACON may be a literal constant. In this case, the operand is indicated as a literal by enclosing it in parentheses. The address constant developed during assembly is the *memory* location of the literal constant, not the constant itself. Figure 22C shows an entry to develop the location of the literal constant. The left-hand address of the constant 1300 is developed.

MACRO-INSTRUCTIONS

MACRO-INSTRUCTIONS may be written into the program in place of the conventional 705 instructions. During assembly, each macro-instruction is replaced by a sequence of component instructions. Parameters of the macro-instruction entry written in the operand column are inserted in the proper locations of the replacing instruction sequence under direction of a control matrix. Each component instruction in the sequence and its corresponding control are stored in the macro-instruction library on the autocoder system tape. The use of macro-instructions in a program does not require an understanding of the mechanics of this process.

However, note that macro-instructions may use the ASU's and the accumulator and, therefore, may affect the sign, zero, high, and equal triggers. The macro-instructions listed in this manual use only ASU's 13, 14, 15 and the accumulator. Some of the macro-instructions assume that ASU 13 is set to 10 and that ASU 14 is set to 4. Refer to the macro-instruction manual for details.

A list of available macro-instructions is furnished in this section. Others may be added by a process explained in the section "Autocoder Library." The writing of macro-instructions is described in the following sections.

Tag

A tag is required only if the macro-instruction is referred to by some other instruction in the program. The tag is usually applied to the first entry in the replacing sequence. The rules for tags are the same as those previously described.

Operation

The operation is the macro-instruction name, e.g., SSGN, SCMP, RSGN, and TYPE.

Numerical, Operand and Comments

The information needed to complete a macro-instruction is written into the numerical, operand, and comments columns of the program sheet. This information must be furnished to the autocoder so that the assembly can complete the component instructions taken from the library.

The numerical column may be used to specify an ASU or other information, such as addresses of components.

Several operands may be needed to complete the instruction. These are written in the operand column and may extend into the comments field. Each operand part is separated by the character ⊡ (lozenge). Each operand part, or a single operand, is written according to the rules associated with operands. For example, the operation MOVE may be used to designate a macro-instruction transmitting a record in memory from one location to another. Three operand parts must be specified to complete the replacing sequence of instructions (Figure 23):

1. Five-character or single-character transmission. If five-character, the accumulator is designated; if single-character, the proper ASU is designated. This information is written into the numerical column.
2. The tag of the record or field to be transmitted. This information is the first part of the operand and is followed by a ⊡ character.
3. The tag of the area which receives the record or field. This information is the second part of the operand and may extend into the comments column.

Listing of Macro-Instructions

The following macro-instructions are available as part of the autocoder system. Complete specifications for each are given in the *Autocoder Macro-Instruction Manual.*

INPUT-OUTPUT MACRO-INSTRUCTIONS

RDTP	Read tape
WRTP	Write tape
DPTP	Dump on tape (WR 01)
RWWTP and WWRTP	Read-while-writing tape
RWWLG and WWRTP	Read-while-writing tape logical (first time RD only)
WRETP	Write erase tape
WRTCP	Write check tape (WR, BSP, RD 01)
WRTM	Write tape mark
BSTP	Backspace tape (specified number of times)
FSTP	Forward space tape (RD 01 specified number of times)
RWDTP	Rewind tape
FWDTP	Forward wind tape (RD 01 forward to next file)
ALTP	Alternate tape units
RDDR	Read drum
WRDR	Write drum
DPDR	Dump on drum (WR 01)
RDCD	Read card
PUNCH	Punch card
DPPCH	Punch 01 card
PRINT	Print under program control
DPPRT	Print 01 under program control
PRNTA	Print under automatic control (single or double)
DPPRA	Print 01 under automatic control
TYPE	Type without checking (turn check indicators off)
DPTYP	Type 01 without checking
TYPCK	Type and check
WREPR	WRE printer (program control)
WREPA	WRE printer (automatic)
WREPN	WRE punch
WRETY	WRE typewriter (without checking)

NOTE: Where applicable, the above macro-instructions provide for inclusion of the appropriate error subroutine, end-of-file transfer address, and restart transfer address. Both latter addresses are optional. If specified, the appropriate transfer will be made; if not, a message will be typed and transfer made to a built-in stop.

The tape instructions are designed for use with the 754 control unit and printer instructions for the 717.

LOGICAL MACRO-INSTRUCTIONS

TRLOW	Transfer low
TRNZ	Transfer non-zero
TRMIN	Transfer minus

LINE	TAG	OPERATION	NUM	OPERAND	COMMENTS
010		MOVE	nn	TAG 1 ⊡ TAG 2 ⊡	
020					

FIGURE 23. MACRO-INSTRUCTION MOVE

TREH	Transfer equal or high
TREL	Transfer equal or low
TRNE	Transfer not equal
FTTR	First-time-only transfer
FTNOP	First-time-only NOP
ALTTR	Alternate transfer (flip-flop)
LOOP	
END	
RPTA	Repeat loop (ASU counter)
SWNOP	Set switch to NOP
RPTM	Repeat loop (memory counter)
SWTR	Set switch to transfer
HLTON	Halt on (if specified alteration switch is on)
HLTOF	Halt off (if specified alteration switch is off)
HLTTR	Halt and transfer
IFXXX	If —, then compare and transfer if specified condition is met
CHKT	Check total
ORDCH	Order check
SEQCH	Sequence check
SETUP	Set ASU's
SSGN	Save sign trigger
SCMP	Save comparison trigger
RSGN	Restore sign trigger
RCMP	Restore comparison trigger
MOVE	Move data (RCV/TMT at either high or low speed)
MOVEC	Move characters (low-speed RCV/TMT the specified number of characters)
MOVEI	Move instruction address (low speed RCV/TMT)
LLL	Load left location (4-digit location of left end of specified field into a specified ASU)
LRL	Load right location
LLL14	Load left location into ASU 14 (4-digit location of left end of specified field, zoned for a specified ASU, into ASU 14)
LRL14	Load right location into ASU 14
DOMIN	Do minus
DOZ	Do zero
DONZ	Do non-zero
DOP	Do plus
DONE	Do non equal
DOEH	Do equal high
DOEL	Do equal low
DOLOW	Do low

FLOATING DECIMAL MACRO-INSTRUCTIONS

FLO	Float
FIX	Fix
FRA	Floating reset add
FRS	Floating reset subtract
FAD	Floating add
FSU	Floating subtract
FMP	Floating multiply
FDV	Floating divide
FAB	Absolute
FST	Floating store
FTP	Floating transfer plus
FTZ	Floating transfer zero
FTM	Floating transfer minus

NOTE: The above macro-instructions make use of a pseudo accumulator. This is a floating decimal accumulator stored in memory as a signed literal work area (+FLOATACCUM).

SUBROUTINES

THE LIBRARY of subroutines forming a part of the autocoder system contains two classes of material. In the first class are the subroutines themselves. These are brief programs which, when executed, perform specific functions, such as taking square root or calculating gross-to-net pay. The other class contains record areas and constant definitions used in defining the data in a program. For example, a rate table may be stored in the library, or the definition of the records employed by an installation may be stored in the library. For convenience, both classes of material are referred to here as "subroutines."

All entries in the library are in autocoder language. Their inclusion in the program being assembled causes them to be processed in the same manner as written entries. They therefore make use of all of the features of the autocoder system.

There are two levels of operation for the incorporation of subroutines. The second level automatically employs the first. The operand in each case is the five-character name of the subroutine to be included.

OPERATION INCL (INCLUDE)

The operation INCL instructs the autocoder to add the label of the subroutine to the list of subroutine requests already encountered, provided that the present

request does not result in duplication. At the end of the assembly, the requested subroutines are located and added to the program. Exit from the main program and linkage to the subroutine must be written by the programmer. Further INCL operations may be encountered in the subroutines being processed. If they are, the additional subroutines are incorporated, provided they are not duplicates.

The operation INCL may appear anywhere in the entry program. The material called for is included in the output at the end of the program. Nothing is inserted in the program at the point occupied by the INCL operation, although the entry appears in the listing.

OTHER OPERATIONS

The following operations automatically include the subroutine and transfer to it subject to certain conditions.

DO	Include subroutine and transfer to it unconditionally.
DOA	Include subroutine and transfer to it on ANY signal.
DOE	Include subroutine and transfer to it on EQUAL.
DOH	Include subroutine and transfer to it on HIGH.
DOS	Include subroutine and transfer to it on SIGNAL.
DOMIN	Include subroutine and transfer to it on MINUS.
DOP	Include subroutine and transfer to it on PLUS.
DONZ	Include subroutine and transfer to it on NON-ZERO.
DOZ	Include subroutine and transfer to it on ZERO.
DONE	Include subroutine and transfer to it on NON-EQUAL.
DOEH	Include subroutine and transfer to it on EQUAL or HIGH.
DOEL	Include subroutine and transfer to it on EQUAL or LOW.
DOLOW	Include subroutine and transfer to it on LOW.

NOTE: The numerical column written in the DO instruction is used to specify the accumulator or an ASU; thus, the appropriate sign and zero indicators are interrogated by the operations DOMIN, DOP, DONZ, and DOZ.

The autocoder provides a linkage to the subroutine requested by generating the sequence INCL, LOD, TR. The operands of the INCL and the TR operation are the same as the operand of the DO operation, that is, the name of the subroutine. The LOD operation places its own location in ASU 14 which is *always assumed to be set to length 4.* If the DO operation is tagged, that tag is applied to the LOD operation; if the DO is not tagged, a tag is generated which consists of □□ followed by a number.

For example, if the programmer writes:

```
INST Y DO SUBROUTINE
```

the autocoder generates and processes:

```
          INCL      SUBROUTINE
INST Y    LOD 14    INST Y
          TR        SUBROUTINE
```

Or, if the DO is not tagged; the autocoder generates and processes:

```
          INCL      SUBROUTINE
□□000n    LOD 14    □□000n
          TR        SUBROUTINE
```

NOTE: A linkage to a subroutine always sets the ASU sign and zero trigger to plus and non-zero, respectively, by execution of the LOD instruction. The settings of the comparison indicator and accumulator sign and zero triggers are not affected.

The DO operation is contained within the autocoder program and is not placed on the library tape. It is not a macro-instruction. One line produced by a DO operation is a TR. The DOA, DOS, DOH and DOE operations are similar and produce, respectively: TRA, TRS, TRH and TRE.

In the case of DOZ, DOP, DONZ, DOMIN, DOEL, DOEH, DOLOW and DONE operations, two-instruction linkage is not suitable. These operations are regular macro-instructions in the library. The programmer using the autocoder need not concern himself with the details.

For example, if the programmer writes:

```
TAG     DOZ     02      SUBROUTINE,
```

the autocoder generates:

```
          INCL            SUBROUTINE
TAG       TRZ     02      TAG + 10
          TR              TAG + 20
          LOD     14      TAG + 10
          TR              SUBROUTINE
```

Listing of Subroutines

The following subroutines are called for and used by the present set of autocoder macro-instructions.

INPUT-OUTPUT ERROR ROUTINES

CDERR	Card read error
CWRER	Check tape-writing error
DRERR	Drum error
PNERR	Punch error
PRERA	Printer error (under program control)
PRERB	Printer error (under automatic control)
RWWER	Read-while-writing tape error
TPERR	Tape error
TYPER	Typewriter error
WRERR	Write erase error
XOFF	Turn check indicators 0901 and 0902 off.

FLOATING DECIMAL ARITHMETIC SUBROUTINES

FAD	Floating add and subtract
FATN	Floating arctan
FDV	Floating divide
FEX	Floating exponential
FIX	Fix
FLN	Floating natural logarithm
FLO	Float
FMP	Floating multiply
FSIN	Floating sine
FSQR	Floating square root

KEY PUNCHING

THE instruction card (Figure 2) is to be key punched from the coding sheet as follows. (Page number and the identification may be duplicated.)

COLUMNS	PUNCHED
1-2	Page number from top of coding sheet. Right justified. Example: Page 2 is punched 02.
3-5	Line number, three characters.
6-15	Tag as written, beginning at left side of field. All letters shown should be punched into consecutive columns unless a "b" separates them. A "b" indicates a column to be spaced over.
16-20	Operation as written, beginning at left side of field.
21-22	Numerical as written, right justified. Some cards may have a field greater than two digits in length; if so, extend to the left, as required, into the operation field.
23-38	Operand. Some operands will be enclosed in parentheses. Punch both right and left parentheses as number sign (#). Some operands may extend beyond the operand column into the comments column. In this case, the comments column is regarded as a continuous extension of the operand column. All letters shown should be punched into consecutive columns unless a "b" separates them. A "b" indicates a column to be spaced over.
39-74	Comments as written.
75-80	Identification on top of coding sheet.

It must be punched in the first card of the program and should be duplicated into the remaining cards.

The following conventions for the writing of certain characters on the coding sheet may prove helpful in avoiding key-punching errors in mixed alphabetic and numerical fields.

WRITTEN CHARACTER	PUNCHED CHARACTER
1	1 (number one)
I	I (letter I)
Ø	0 (zero)
Ø	O (letter O)
b	blank
(	#
)	#
&	12 punch
+	12 punch
—	11 punch
ǂ	12-5-8 punch
†	0-2-8 punch
±	12-0 punch
∓	11-0 punch
2	2 (number 2)
Z	Z (letter Z)

OPERATING INSTRUCTIONS

THE FOLLOWING sections list the operating instructions for 705 operation of the autocoder. Operating instructions will be distributed with the autocoder system deck. These operating instructions will supersede this entire section.

Program Assembly

1. The assembly uses the following machine units:

0200	Autocoder system tape
0201	Expanded entries
0202	Instruction input-output program listings
0203	Tags
0205	Tags, literals and controls
0206	Output program
	OPTIONAL UNITS
0100	Instruction card input
0300	Program card output
0400	Program listing output
(0264)	Program listing output, high-speed printer

If the autocoder system is set up for use with TRC's, tape unit addresses are 0600, 0601, 0602, and so on.

If the system is set up with 760 and/or TRC's, the listing is also written on 0607 (0207), with record marks replaced by blanks and group marks replaced by lozenges (□). This latter tape is to be used for auxiliary listing on the high-speed printer, while 0202 (0602) is to be used for reassembly.

2. Set alteration switches:

0911	ON	Tape input (0202) for instruction records.
	OFF	No tape input (0202).
0912	ON	Instruction card record input.
	OFF	No instruction card input.
0915	ON	Direct output program listing on the printer. (Program listing is always written on tape 0202.)
	OFF	Output program listing on tape 0202 only.
0916	ON	Direct program card output. (The assembled program is always written on 0206.)
	OFF	Output program on 0206 only.

3. Check switches

0902	PROGRAM
0903	PROGRAM

All others to AUTOMATIC.

4. Manually give the instructions:

SEL	(2)	0200
RD	(Y)	0000

5. Press RESET and START.

The 705 halts 6996 following a message describing the alteration switch settings the console operator has selected. If the settings are correct, press START. If the settings are incorrect, change the alteration switches, RESET and START. The alteration switch setting routine will be repeated.

The original input tape (card image or previous listing) will be destroyed by the new listing tape produced by the assembly on tape unit 0202. To conserve the original input tape, it is necessary to file-protect the input tape originally, changing the tape setting in phase II to a non-file-protected 0202.

If the output program is to be loaded from tape 0206 directly, when it is placed in operation, the first instruction on tape 0206 must be modified manually before proceeding. This can be accomplished as follows:

1. Press INSTRUCT key.
2. SEL (2) 0206*
3. RD (Y) 0000
4. Set memory address selector keys to 0002.
5. Press STORE key.
6. Depress 206* on keyboard.
7. Press RESET and START.

*or the address of the tape unit containing the program, if this was changed after assembly.

Restart at Beginning of Phase II

This operation is used only if assembly was not completed after completion of phase I.

1. Mount tapes removed at end of phase I on the following tape units:

0200	Autocoder system
*0201	Expanded entries
0202	Scratch
*0203	Tag table
*0205	Tags, literals, and controls
*0206	Scratch

0207 Scratch (760 only)

*All of these tapes must have been saved from phase I.

2. Set alteration switches:

0914	ON	Restart phase II.
0915	ON	Direct program listing on printer. (Program listing is always written on 0202.)
	OFF	Listing on 0202 only.
0916	ON	Direct program card output. (The assembled program is always written on 0206.)
	OFF	Output program on 0206 only.

3. Set check switches:

0902 PROGRAM
0903 PROGRAM
All others to AUTOMATIC.

4. Manually give the instructions:

SEL	(2)	0200
RD	(Y)	0000

5. Press RESET and START.

Complete operating instructions, with a list of messages and program stops, are furnished with each autocoder system.

REASSEMBLY

REASSEMBLY is a function by which a program on tape in autocoder language is processed while the autocoder assembly incorporates corrections and revisions from instruction cards. It is intended to save machine time (or card-to-tape conversion) during program testing or when up-dating an established program. During reassembly, entries may be inserted, substituted or deleted from the original output program. The result is a new output program incorporating the desired changes. The optional output listing and program deck also contain the up-dated program.

Each entry is identified for reassembly purposes by its page and line number. The original program on tape is assumed to be in page and line number sequence as is the set of revision cards in the card reader.

NOTE: During assembly, if two or more consecutive entries bear the same page and line number, all but the first entry will be listed with blank page and line numbers.

The tape may contain either the original 80-character card records resulting from a card-to-tape conversion or the 120-character printer records on the listing tape produced by the autocoder during a previous run. The autocoder determines which type is to be reassembled by analysis of the first record.

Once the original program has been processed by the autocoder, it is necessary to preserve only the listing tape because it provides all of the information necessary to effect a reassembly. Hence, a program may be brought from preliminary to final form through a succession of listing tapes. Also, the listing tape has the desirable quality of being already edited for printing.

Insertion

Card entries will be inserted in the program if they have page and line numbers which cause them to fall between successive tape entries. Alphabetic punching of line numbers may be useful in providing additional insertion space.

Substitution

A match between card-entry and tape-entry page and line number causes the card entry to be substituted for the tape entry.

Deletion

A card bearing the operation code DELET and the page and line number of a tape entry will cause that entry to be deleted from the output program. If the operand field of the delete card specifies a second page and line number (higher in the sequence), all the tape entries between the lower and upper page and line numbers (inclusively) will be deleted from the output.

Program Reassembly

1. Machine units used are the same as for assembly. Load the input tape (previous listing tape or card image tape) and/or cards in card hopper.

2. Set alteration switches:

0911	ON	Previously assembled tape input (on 0202), either program listing or program tape.
	OFF	No tape input.
0912	ON	Instruction card input (correction cards).
	OFF	No card input.
0915	ON	Listing directly on printer (listing always on 0202).
	OFF	Listing on 0202 only.
0916	ON	Direct program card output (always on 0206).
	OFF	Program on 0206 only.

3. Set check switches:
   0902 PROGRAM
   0903 PROGRAM
   All others to AUTOMATIC

4. Manually give the instructions:
   SEL (2) 0200
   RD (Y) 0000

5. Press RESET and START.

The 705 halts 6996 following a message describing the alteration switch settings.

If the settings are *correct,* press START. If the settings are *incorrect,* change the alteration switch settings; press RESET and START. The alteration switch settings routine is repeated.

The original input tape (card image of previous listing) is destroyed by the new listing tape produced by the assembly on tape unit 0202.

If the output program is to be loaded directly from tape, 0206, when it is placed in operation, the first instruction on the tape must be modified manually before proceeding.

1. Press INSTRUCT key
2. SEL (2) 0206*
3. RD (Y) 0000
4. Set memory address selector keys to 0002.
5. Press STORE key.
6. Depress 206* on keyboard.
7. Press RESET and START.

*or the address of the tape unit containing the program card record, if this was changed after assembly.

NEW OPERATION, ACON 4

The operation ACON 4 (address constant four digits) is another type of address constant as described on page 24. It is used exactly in the same manner as RACON, but produces just the four digit machine address as a constant; it is not preceded by the character A. It will be placed in the next four available memory positions, regardless of whether the location ends in a 4 or 9.

PROGRAMMING NOTES

A constant of a single & (ampersand) or - (hyphen) may be created in memory by defining a constant of length one with the sign in the first position of the operand field.

Single character literals of the form &, -, #, (ampersand, hyphen, or number sign) are acceptable entries to produce the desired constant.

AUTOCODER OUTPUT

THE AUTOCODER produces as output a program listing and a program deck. This deck consists of loading instructions and the 705-coded result of the autocoder assembly.

ORGANIZATION OF MEMORY

AFTER loading the program deck in preparation for actual operation, the data are arranged as follows:

1. The first information is the loading program, occupying 80 character locations (Figure 24).

2. The first program entry follows immediately after the loading program, unless otherwise specified by an LASN or SASN entry with an actual operand.

3. Succeeding entries (instructions, constants, record areas) follow in order, unless otherwise specified by LASN, or SASN entries or by actual or descriptive operands on the "define" line of an area of constants or record storage.

4. Subroutines, if any, follow the final program entry.

5. After the subroutines (or the final program entry if there are no subroutines) there follows: (a) a blank, (b) signed literals, and (c) unsigned literals.

6. Space is left in memory between entries whenever: (a) instructions are shifted forward in order to place the operation character on a 0 or 5 location, and (b) areas are shifted as directed by an entry in the numerical column of a "define" operation.

Type-outs occur during assembly if there is an assignment of locations above 20,000 or 40,000.

INSTR. LOCATION	INSTRUCTION OPER.	INSTRUCTION ADDRESS	STOR. CODE	ACCUMULATOR 00	SIGN	AUXILIARY STORAGE 01-15	SIGN	
0004	SEL	0100						
0009	NOP	0018						
0014	SET	0002	6					
0019	SET	0004	7					
0024	RD	0080						
0029	TRA	0079						
0034	LOD	0094	6					
0039	TRZ	0099	6					
0044	UNL	0059	6					
0049	RCV	0061						
0054	TMT	0089	7					
0059	SET	00--	8					
0064	RCV							
0069	TMT	0095	8					
0074	TR	0004						
0079	HLT	100#						

FIGURE 24. LOAD PROGRAM

OUTPUT PROGRAM

THE OUTPUT produced by the autocoder may be either on IBM cards or on tape as individual card records. Output consists of:

1. A single-card loading routine.
2. The output program on program cards.
3. A 00 control card which is always placed at the end of the program deck to transfer to the first instruction of the program. Other control cards may be placed in the deck.

The loading routine reads load cards and transmits their contents to the proper locations in memory.

Program cards are punched as follows:

Cols. 1-6	Identification
Cols. 7-9	Serial number
Cols. 10-13	Initial address
Cols. 14-15	Number of columns
Cols. 16-80	Instructions and data

The cards are identical with symbolic assembly output (*Program Brief 6*).

If a load card contains the quantity 00 in columns 14 and 15, control is transferred by the loading routine to column 20 of the load card which is assumed to contain an instruction. The last card produced as output by the autocoder is a 00 card that establishes certain ASU settings and transfers control to the *first instruction* (as opposed to other types of entries) in the program.

As a consequence of the loading program itself, and of the instructions contained on the 00 card, the following ASU settings and contents are established. (In this list the asterisk indicates a setting essential to the functioning of macro-instructions and subroutines produced by the autocoder. The other settings are of convenience in program preparation.)

ASU	LENGTH	CONTENT
1	1	—
2	2	—
3	3	—
4	4	—
5	5	—
*13	10	—
*14	4	—

AUTOCODER LIBRARY

THE AUTOCODER library is divided into two sections, macro-instructions and subroutines. Each section is arranged in sequence by the names of the macro-instructions and subroutines. Use of an entry in the library does not alter data in any manner; information remains unchanged on the autocoder system tape.

Material in the library may be added, deleted, or substituted as described in "Librarian." In this manner, an installation may build its own library to include information which may be adapted to particular applications.

WRITING MACRO-INSTRUCTIONS

REFER to this section only when it becomes necessary to prepare supplementary instructions to meet the requirements of an individual installation. An understanding of the process of preparing new macro-instructions to be added to the autocoder is not a requirement of program writing, assembly, or use of the autocoder system.

The library of the autocoder is intended to be as general in application as possible. Typical of routines available at present are address arithmetic, logical functions, floating point arithmetic, common mathematical functions, and input-output routines. The library contains both macro-instructions and subroutines. Some macro-instructions call for a subroutine and some subroutines use other subroutines as well as macro-instructions. A single reference to the library in an autocoder program may, therefore, lead to the selection of several routines from the library. The programmer need be concerned only with the library reference which initiates the chain; the rest is automatic.

Form

The form in which the macro-instruction is written is determined by the need it must fulfill. A macro-instruction may have a single or multiple address. Its numerical field may be used for an ASU designation or a tape unit specification or it may have other significance convenient to its function. The format is variable, depending upon the use for which the instruction is designed. Each macro-instruction in the library is described in detail by exact specifications telling how it is to be written and what data are to be supplied on the program sheet.

A macro-instruction represents a sequence of other 705 and/or autocoder operations grouped together for the convenience of the programmer. Any given programmed function also can be written using a number of conventional 705 instructions and other autocoder operations. It is usually more economical of programming effort, however, to write a single macro-instruction instead.

Component Instructions

In the following explanation, the term "macro-instruction" refers to the written program entry. This entry causes the autocoder assembly to refer to the library for further information. Instructions taken from the library and included in the output program are referred to as component instructions.

The components in the library are all autocoder operations, i.e., 705 instructions, definitions, address constants, and DO *operations*. A single macro-instruction may produce up to 20 components. Only the last component of the series may be another macro-instruction. This feature may be used to secure more than 20 components. (For this purpose, DO, DOA, DOE, DOH and DOS are not considered macro-instructions; they may be placed anywhere in the series.)

The component instructions, as they are stored in the library, are only partially completed. The autocoder assembly uses the information supplied by the macro-instruction entry on the program sheet to fill in the missing parts. For example, the macro-instruction MOVE may be used to transfer information from a designated location in memory to some other location. (Refer to specification of MOVE.)

Several items of information are furnished by the macro-instruction:

(1) the tag of the received area, (2) the tag of the area to be transmitted, and
(3) the ASU designation, if serial transmission is to be used.

The two components of the MOVE macro-instruction contained in the library are the instructions RCV and TMT without addresses or ASU designation. The

tags of the areas to be moved are translated by the assembly to actual addresses, the proper ASU is specified, and the completed instructions are placed in the main program.

Control Matrix

Each partial instruction in the library has an associated five-character code field. This code tells the autocoder assembly how each field of the component is to be treated. The five-character code is called a control matrix and each position of the code is referred to as a cell within the matrix. Reading from left to right, the cells correspond to the columns of the autocoder program sheet except the fifth cell (Figure 25).

A character entry in the first cell causes the insertion of up to ten characters in the tag field of the component. An entry in the second cell causes the insertion of up to five characters in the operation field of the component. An entry in the third cell causes the insertion of up to two characters in the numerical field; an entry in the fourth cell, up to 16 characters in the operand field; an entry in the fifth cell causes the insertion of up to 52 characters in the operand field of the component.

Should a component in a macro-instruction refer to another component in the series, one of three tags may be generated by the autocoder for the component. The three possibilities are tags of eight characters in length with either one lozenge, three lozenges, or four lozenges as the initial characters. Only one of each can be generated for the component entries of a macro-instruction. One component may also be tagged with the tag given to the macro-instruction. The use of codes A, E, F, W, X, Y, and Z in the matrix determines which tags are generated. (It will be noted that tags with two lozenges are generated for DO instructions. See "Subroutines" in "Autocoder Programming.")

Figure 26 is a list of permissible character entries into the control matrix. The control functions of each character, through which the autocoder is told how to complete each component, are described. The character used determines the place from which the information is to be obtained. The number of characters moved is determined as the lesser of two limits. No more characters are moved than the source can supply nor more than the destination can accept. These limits are 10 for tag, 5 for operation, 2 for

Matrix Code	Description
blank	No effect
A	Insert tag found in macro-instruction
B	Insert operation found in macro-instruction
C	Insert numerical found in macro-instruction
D	Insert operand (max. 10 characters) found in macro-instruction.
E	Insert ¤ quantity as generated by the Autocoder.
F	Examine tag found in macro-instruction. If blank, consider as E; if not blank, consider as A.
J	**Last component instruction. May be in any column.**
K	Insert numerical found in macro-instruction into positions 3 and 4 of operand. May be in any column of the matrix.
L	Insert numerical found in macro-instruction into positions 3 and 4 of operand. SUB (-0) to add leading zero if necessary.
M	Insert operand (max. 16 characters) found in macro-instruction. Check for character adjustment in macro-instruction and component instruction, using the algebraic sum as the character adjustment in the resulting 705 instruction.
P	Insert operand (max. 52 characters) found in macro-instruction.
W	Insert ¤¤¤ quantity as generated by Autocoder.
X	Examine tag found in macro-instruction. If blank, consider as W; if not blank, consider as A.
Y	Insert ¤¤¤¤ quantity as generated by Autocoder.
Z	Examine tag found in macro-instruction. If blank, consider as Y; if not blank, consider as A.
1	Search for the first lozenge in the macro-instruction and insert in the component all the characters preceding it in the operand.
2	Search for the second lozenge in the macro-instruction and insert in the component the characters between the 1st and 2nd lozenge.
3	Same - between 2nd and 3rd lozenge.
4	Same - between 3rd and 4th lozenge.
5	Same - between 4th and 5th lozenge.
6	Same - between 5th and 6th lozenge.
7	Same - between 6th and 7th lozenge.
8	Same - between 7th and 8th lozenge.
9	Same - between 8th and 9th lozenge.

FIGURE 26. TABLE OF MATRIX CODES

TAG	OPERATION	NUM	OPERAND	COMMENTS

FIGURE 25. MATRIX ARRANGEMENT

numerical, 16 for operand, and 52 for operand when the control character is in the fifth column.

When a component is complete, it is further processed by the autocoder in the same manner as other entries from the instruction cards.

Macro-Instruction in Preparation

In the preparation of a new macro-instruction to be added to the library, its functions must be properly defined. For the macro-instruction MOVE, it is known that two instructions, RCV and TMT, always serve to transfer a memory area from one location to another. Therefore, the component instructions consist of these two instructions only. It is also known that three separate pieces of information must be supplied from the macro-instruction. These are the addresses for the receive and transmit instructions and the ASU designation to specify serial or five-character transmission. The specification for MOVE calls for the macro-instruction to be written as shown in Figure 23. Tag 1 represents the location of the "from" area, while tag 2 represents the location of the "to" area and *nn* represents the ASU designation from 00 to 15. The code MOVE has been decided upon as the mnemonic name of the macro-instruction.

The next consideration concerns the component instructions themselves and the arrangement of their associated matrices. The components are written on the autocoder program sheet in normal fashion as shown in Figure 27. Note that the instructions RCV and TMT are written in the operation column. The control matrix is written in the first five positions of the comments column. The first cell entry contains an A to allow tagging of the RCV instruction at the option of the user; the second entry is blank. The third character is C, which tells the assembly to insert the numerical designation into the instruction. The fourth character of the RCV instruction is a 2, which searches for the second lozenge in the macro-instruction operand and inserts the characters preceding it into the component operand. This is the tag of the "to" area. The fourth character of the TMT component is a 1, which inserts the characters preceding the first lozenge of the macro-instruction operand into the operand of the component instruction. This is the tag of the "from" area. The fifth cell of the matrix contains J to signify that this is the last instruction in the sequence.

Figure 28 is a listing of the components for certain macro-instructions in the autocoder library. It should serve to indicate some ways to use the matrix characters.

KEY PUNCHING COMPONENT INSTRUCTIONS

COMPONENT instructions are key punched from the autocoder program sheet, one instruction per card, using the autocoder instruction cards shown in Figure 2. Instructions are punched as described in the section "Key Punching Instruction Cards," except that the control matrix is punched into the first five positions of the comments field. The five-character mnemonic identification of the macro-instruction as written after IDENT on the top of the coding sheet is punched into each card, preceded by the letter M, into columns 75-80. Only one series of components may be punched from a single program sheet, because each series must have its own unique identification.

A header card is also punched for each macro-instruction as follows:

Operation field (cols. 16-20)	"INSER"
Operand field (cols. 23-26)	Number of components.
Identification field (cols. 75-80)	Letter M, followed by mnemonic identification.

The completed macro-instruction is placed in the library by a special processing run on the 705 as described in the "Librarian" section.

LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
010		RCV			AbC2b
020		TMT			bbC1J
030					

FIGURE 27. MACRO-INSTRUCTION COMPONENTS, MOVE

LNE	OP	NU	OPERAND	COMMENTS	IDENT
			ALTP		
20	LOD	14	F	1	MALTP
40	RAD	15	&00020	F	MALTP
50	UNL	15		1	MALTP
60	ST	14	&00020	F	MALTP
70	AACON	&		2J	MALTP
			ALTTR		
30	TR		&00015F	F	MALTTR
40	SGN	15	-00004	F	MALTTR
50	TR		&00030	F	MALTTR
60	SGN	15	-00004	F	MALTTR
70	ADM	15	-00004	F	MALTTR
80	TR			PJ	MALTTR
			BSTP		
20	SEL		02 F	L	MBSTP
30	RAD	15	&00030	F	MBSTP
40	BSP				MBSTP
50	SUB	15	#&1#		MBSTP
60	TRZ	15	&00035	F	MBSTP
70	TR		&00010	F	MBSTP
80	AACON	&		DJ	MBSTP
			CHKT		
20	SET		00 A	L	MCHKT
30	LOD			1	MCHKT
40	SUB		#&0#		MCHKT
50	TR			E	MCHKT
60	DCON		2		MCHKT
70		05	00000		MCHKT
80			&0 C		MCHKT
90	ADM		E	2J	MCHKT
			DOEH		
20	TRE		&00015F	F	MDOEH
30	TRH		&00015	F	MDOEH
40	TR		&00025	F	MDOEH
50	DO			PJ	MDOEH
			DOEL		
20	TRE		&00010F	F	MDOEL
30	TRH		&00020	F	MDOEL
40	DO			PJ	MDOEL
			DOLOW		
20	TRE		&00020F	F	MDOLOW
30	TRH		&00020	F	MDOLOW
40	DO			PJ	MDOLOW

FIGURE 28. AUTOCODER LIBRARY MACRO-INSTRUCTIONS

GENERAL CONSIDERATIONS

THE FOLLOWING general considerations should be noted in preparing macro-instruction components and in using macro-instructions, unless otherwise specified.

1. Tagging of macro-instructions is optional.

2. Component instructions do not have tags. Tags, as required, must be supplied through the use of the control matrix.

3. Macro-instructions listed in this manual leave the sign trigger of the ASU's either undisturbed or set plus. Exceptions are noted in the macro-instruction manual.

4. Macro-instructions listed in this manual leave the zero trigger of the ASU's either undisturbed or turned off. Exceptions are noted.

5. Macro-instructions do not alter the condition of the high and equal triggers, unless otherwise specified.

6. Some of the macro-instructions and the subroutines they include make use of ASU's 13, 14 and 15, subject to considerations 3, 4, and 5 above and also subject to the provision that they find and leave ASU 13 set to length 10 and ASU 14 set to length 4. The floating arithmetic routines also make free use of the accumulator. Any additional ASU requirements are noted.

7. A character adjustment, if needed, must be placed in positions 11 through 16 of the operand column of the components.

8. Literals for working storage should contain at least one non-numerical character to eliminate the possibility of conflict with literals generated by component instructions.

WRITING SUBROUTINES

A detailed knowledge of linked subroutines is not necessary for their proper use or the proper use of macro-instructions that employ them. However, when it is desired to write new subroutines, or to make changes in those already existing, to fit the needs of a particular installation, then it is necessary to know linked subroutines in some detail. It is very important here to understand the relationship between the format of an input-output macro-instruction, and the error subroutine which operates on it in case of error or end-of-file conditions.

There are three basic considerations which determine whether a series of 705 instructions should take the form of an inline sequence (macro-instruction) or of a linked subroutine: space, time and flexibility. A linked sub-routine takes more time to execute, but saves space in direct proportion to the number of times that the same series of instructions may be encountered at different points within a given program. Macro-instructions (inline sequences) are flexible, in that addresses can be specified by the programmer while subroutines must be of a fixed form and can be assembled only in that form. The programmer has no method of modifying a subroutine in the autocoder assembly process.

Figure 29 of the macro-instruction TYPE and the linked subroutine XOFF, illustrate the linkage to, and from, a subroutine. A linked subroutine is used in this case to save space.

The linkage here is the simplest that is possible. Following the execution of the WR instruction, the location of the LOD instruction is loaded in ASU 14 (alpha load alpha). If the ANY indicator is on, a transfer is made to the subroutine XOFF. Here the location is unloaded from ASU 14 into the last instruction in the subroutine to be executed, which is a TR instruction. A constant of ten is added to this instruction so that the TR instruction will cause a return to the correct place in the program after turning off the indicators.

The error subroutines associated with the input-output macro-instructions require that more than an exit back be provided. Typically, the unit selected, the reading or writing address, the end-of-the-file transfer address, and the restart transfer address must be moved from the main routine into instructions in the subroutine so that the subroutine can carry out its assigned functions. The tape error subroutine (TPERR), which is listed in Figure 30 is of this type.

The linkage (from line 001 to line 054) consists of the instructions necessary to initialize the basic subroutine (from line 055 to line 089). Following a TRA from the main routine (see Figure 30, macro-instructions RDTP, page 01, line 010, and WRTP, page 01, line 190) to TPERR, the location is unloaded from ASU 14 into certain load instructions which are then modified by either increments or decrements so that these load instructions can move the necessary addresses from the main routine into the subroutine.

For example, at line 004 the location is unloaded into a LOD instruction at line 012. At line 011 a

LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
010	T1	TYPE		MESSAGE	Coded by programmer
020					
030	T1	SEL		500	Instructions generated
040		WR		MESSAGE	
050		DOA		XOFF	Generates next three entries
060		INCL		XOFF	
070	␣␣n	LOD	14	␣␣n	
080		TRA		XOFF	
090					
100					
110					
120					
130		SUBROUTINE XOFF			
140					
150	XOFF	UNL	14	XOFF2	UNLOAD LOCATION INTO EXIT
160		LOD	14	(0010)	
170		ADM	14	XOFF2	ADD TEN TO EXIT
180		SEL		901	
190		TRS		XOFF1	TURN OFF 901
200	XOFF1	SEL		902	
210		TRS		XOFF2	TURN OFF 902
220	XOFF2	TR			EXIT BACK TO MAIN ROUTINE
230					
240					

FIGURE 29. EXAMPLE OF LINKED SUBROUTINE

decrement of ten (I990, 40,000's complement of ten) is added to the LOD instruction. Therefore, when this LOD instruction is executed, it has the proper address and proper ASU zoning to load the SEL address into ASU 14.

In a similar manner, the other parameters needed are moved to the subroutine. Subroutines may call for other subroutines. For example, TPERR, in turn, alpha-loads-alpha and transfers to XOFF after typing messages (line 071).

The DO operation may be character adjusted, in which case the named subroutine will be included and transfer will be made to some other point than the normal entry line of the subroutine. Also the INCL operation honors only the first five characters which are the name of the subroutine. If a DO, or an INCL, operation contains the tag of an entry within the subroutine as the operand, and if the first five characters of all the tags within a subroutine have been planned to be precisely the name of the subroutine, including blanks, if necessary, these operations will cause the inclusion of the subroutine and transfer to the tagged entry within the subroutine. Ordinarily this operation will be useful when a single subroutine is devised to carry out two similar functions by having two entry points. An example of this is the subroutine FLOATING ADD which has an additional entry to accommodate FLOATING SUBTRACT operations.

AUTOCODER SERIAL TABLE SEARCH

Figure 30 illustrates the use of macro-instructions in writing a program to determine total cost (unit cost $\times$ item quantity $=$ total cost). Item quantity is located in each record. The unit cost for the item in each record is determined by performing a serial table search. The table is entered in memory as a constant and is made up of 20 two-digit codes with their associated four-digit unit costs. The item code and its associated cost are separated by a blank. The item codes are in order according to frequency of use.

The input record is on tape #0202. Entries in the item code and item quantity fields are already in the record on tape. Entries for the unit cost and total cost fields are supplied and developed by the program.

The instructions which the programmer wrote are shown in Figure 31. All other instructions were generated by the autocoder system. Macro-instructions and the instructions linking the main program to subroutines are made to stand out by over-printing.

The table search routine (Figure 30) is initialized for each input record by the macro-instruction MOVEI, located at 01040. This macro-instruction moves the address portion from the instruction specified by the first operand to the instruction specified by the second operand by producing a receive-and-transmit instruction via ASU 14. A tag may be supplied in the tag field (see entry 01110), or it may be left blank as in entry 01040.

The initial address of the compare instruction (01060) is modified after each comparison by the entry *ASU04 in the comments field. This entry produces the add memory instruction immediately following entry 01060. This ADM steps up the address of the compare instruction to the next item code in the table. If the equal trigger is on at entry 01070 the address of the unit cost needed for computation is two positions less than the address in entry 01060. Consequently, this address is moved to entry 01140. The address is then decreased by two by adding the 40,000's complement of two (I998) to it.

The total cost is then computed and a new record is brought in.

```
PG LINE    TAG          OP       NUM    OPERAND                PROGRAM TBL SH COMMENTS          LOC OP ASU ADDRESS
01 000                  TITLE                                  SERIAL TABLE SEARCH

01 010 READ RCD        RDTP      2      RCD AREA¤EOJ RT¤¤
       READ RCD        SEL              0202                                                    00164 2     00202 0202
                       RD               RCD AREA                                                00169 Y     00400 0400
                       DOA              TPERR
                       INCL             TPERR
       ¤¤000001        LOD       14     ¤¤000001                                                00174 8 14  00174 0AP4
                       TRA       14     TPERR                                                   00179 I 14  00574 0EP4
                       RACON            EOJ RT                                                  00184 A     00334 0334
                       RACON                                                                    00189 A
                       INCL             XOFF
01 020                 RAD       4      #&0007#          TO STEP COMPARE ADDRESS                00194 H  4  01059 1 59
01 030                 LRL14            LAST CODE&7
       ¤0000002        LOD       14     ¤0000002  &5                                           00199 8 14  00204 0B-4
                       RACON            LAST CODE &7                                            00204 A     00562 0562
01 040                 MOVEI            BASIC ADDR¤TBL SEARCH¤
       ¤0000003        RCV       14     TBL SEARCH&1                                            00209 U 14  00221 0BK1
                       TMT       14     BASIC ADDR&1                                            00214 9 14  00566 0EO6
01 050                 RAD              ITEM CODE                                               00219 H     00402 0402
01 060 TBL SEARCH      CMP                               *ASU04                                 00224 4
                       ADM       4      @00224                                                  00229 6  4  00224 0S24
01 070                 TRE              ITEM FOUND                                              00234 L     00254 0254
01 080                 CMP       14     TBL SEARCH       IS THIS LAST ITEM                      00239 4 14  00224 0BK4
01 090                 TRE              NO CODE          YES                                    00244 L     00394 0394
01 100                 TR               TBL SEARCH       CONTINUE TABLE SEARCH                  00249 1     00224 0224

01 110 ITEM FOUND      MOVEI            TBL SEARCH¤COMPUTE¤
       ITEM FOUND      RCV       14     COMPUTE   &1                                            00254 U 14  00271 0BP1
                       TMT       14     TBL SEARCH&1                                            00259 9 14  00221 0BK1
01 120                 LOD       14     #1998#           TO DECREMENT BY 2                      00264 8 14  01068 1&08
01 130                 ADM       14     COMPUTE                                                 00269 6 14  00274 0BP4
01 140 COMPUTE         RAD                               UNIT COST                              00274 H
01 150                 ST               UNIT COST                                               00279 F     00406 0406
01 160                 MPY              ITEM QTY                                                00284 V     00411 0411
01 170                 RND              1                                                       00289 E     00001 0001
01 180                 ST               TOTAL COST                                              00294 F     00419 0419
01 190                 WRTP      1      RCD AREA¤¤¤
                       SEL              0201                                                    00299 2     00201 0201
                       WR               RCD AREA                                                00304 R     00400 0400
                       DOA              TPERR
       ¤¤000002        LOD       14     ¤¤000002                                                00309 8 14  00309 0C-9
                       TRA       14     TPERR                                                   00314 I 14  00574 0EP4
                       RACON                                                                    00319 A
                       RACON                                                                    00324 A
01 200                 TR               READ RCD                                                00329 1     00164 0164

01 210 EOJ RT          WRTM      1
       EOJ RT          SEL              0201                                                    00334 2     00201 0201
                       WTM                                                                      00339 3     00001 0001
                       DOA              TPERR
       ¤¤000003        LOD       14     ¤¤000003                                                00344 8 14  00344 0CM4
                       TRA       14     TPERR                                                   00349 I 14  00574 0EP4
                       RACON            EOJ RT    &30                                           00354 A     00364 0364
                       RACON                                                                    00359 A
01 220                 RWD                                                                      00364 3     00002 0002
01 230                 TYPE             #END OF JOB¤#
                       SEL              0500                                                    00369 2     00500 0500
                       WR               #END OF JOB¤#                                           00374 R     01069 1069
                       DOA              XOFF
       ¤¤000004        LOD       14     ¤¤000004                                                00379 8 14  00379 0CP9
                       TRA       14     XOFF                                                    00384 I 14  01019 1&J9
01 240                 HLT              9999                                                    00389 J     09999 9999
01 250 NO CODE         HLT              1                                                       00394 J     00001 0001
01 260                 TR               NO CODE                                                 00399 1     00394 0394

02 010                 DRCD                                                                     00400
02 020 RCD AREA                 1       &                                                       00400
02 030 ITEM CODE                2       &                                                       00402
02 040 UNIT COST                4       &                                                       00406
02 050 ITEM QTY                 5       &                                                       00411
02 060 TOTAL COST               8       &                                                       00419
```

FIGURE 30A. 705 LISTING OF SERIAL TABLE SEARCH

```
PG LINE   TAG          OP      NUM    OPERAND          PROGRAM TBL SH COMMENTS                    LOC OP ASU ADDRESS
02 070                 DCON                                                                      00420
02 080                            1   ¤                                                         00420
02 090 FIRST CODE                 2   03               TABLE                                    00422
02 100                            1                                                            00423
02 110                            4   047H             UNIT COST FIELD                          00427
02 120                           49   02 124E01 112G17 074A05 299C06 151A12 038G08 293I         00476
02 130                           49   09 300C19 290C11 281I07 158D13 366A20 105D15 102B         00525
02 140                           28   04 002G18 012F16 113C10 024G                              00553
02 150 LAST CODE                  2   14                                                       00555
02 160                            5    106D                                                    00560
02 170 BASIC ADDR RACON                FIRST CODE                                               00569 A    00422 0422

SU LNE  FOLLOWING   ARE               SUBROUTINES
01 001 TPERR       SET     15   1                TAPE ERROR SUBROUTINE                        00574 B 15 00001 0&&1
01 002             LOD     15   TPERR K1  -4     A                                            00579 8 15 00825 0HB5
01 003             UNL     15   TPERR10.1 -4     MAKE NOP                                     00584 7 15 00850 0HE0
01 004             UNL     14   TPERR1                                                        00589 7 14 00629 0FK9
01 005             UNL     14   TPERR2                                                        00594 7 14 00674 0FP4
01 006             UNL     14   TPERR3                                                        00599 7 14 00704 0G-4
01 007             UNL     14   TPERR4                                                        00604 7 14 00719 0GJ9
01 008             UNL     14   TPERR6                                                        00609 7 14 00749 0GM9
01 009             UNL     14   TPERR13                                                       00614 7 14 00899 0HR9
01 010             LOD     14   #I990#                                                        00619 8 14 01083 1&Q3
01 011 TPERR1.1    ADM     14   TPERR1                                                        00624 6 14 00629 0FK9
01 012 TPERR1      LOD     14                    TAPE UNIT NUMBER                             00629 8 14
01 013             UNL     14   TPERR10                                                       00634 7 14 00844 0HM4
01 014             UNL     14   #EOF X XXXX¤#-1                                               00639 7 14 01093 1&R3
01 015             UNL     14   #901 X XXXX¤#-1                                               00644 7 14 01104 1A-4
01 016             UNL     14   #902 X XXXX¤#-1                                               00649 7 14 01115 1AJ5
01 017             UNL     14   TPERR24                                                       00654 7 14 00889 0HQ9
01 018             LOD     14   #I9Z1#                                                        00659 8 14 01120 1AK0
01 019             ADM     14   TPERR2                                                        00664 6 14 00674 0FP4
01 020             ADM     14   TPERR3                                                        00669 6 14 00704 0G-4
01 021 TPERR2      LOD     15                    RD OR WR OP CODE                             00674 8 15
01 022             UNL     15   #EOF X XXXX¤#-6                                               00679 7 15 01088 1&H8
01 023             UNL     15   #901 X XXXX¤#-6                                               00684 7 15 01099 1&I9
01 024             UNL     15   #902 X XXXX¤#-6                                               00689 7 15 01110 1AA0
01 025             RAD     15   #&00000#         INITIALIZE COUNTER                           00694 H 15 01064 1&F4
01 026             RCV     15   TPERR11                                                       00699 U 15 00855 0HE5
01 027 TPERR3      TMT     15                    RD OR WR INSTR                               00704 9 15
01 028             LOD     14   #0010#                                                        00709 8 14 01124 1AK4
01 029             ADM     14   TPERR4                                                        00714 6 14 00719 0GJ9
01 030 TPERR4      LOD     14                    EOF ADDR                                     00719 8 14
01 031             CMP     14   #    #                                                        00724 4 14 01128 1AK8
01 032             TRE     14   TPERR8           NO EOF ADDR SPECIFIED                        00729 L 14 00789 0GQ9
01 033             UNL     14   TPERR26                                                       00734 7 14 00909 0I-9
01 034 TPERR5      LOD     14   #C015#                                                        00739 8 14 01132 1AL2
01 035             ADM     14   TPERR6                                                        00744 6 14 00749 0GM9
01 036 TPERR6      LOD     14                    RESTART ADDR                                 00749 8 14
01 037             CMP     14   #    #                                                        00754 4 14 01128 1AK8
01 038             TRE     14   TPERR9           NO RESTART ADDR SPECIFIED                    00759 L 14 00804 0H-4
01 039             UNL     14   TPERR18                                                       00764 7 14 00964 0I04
01 040             UNL     14   TPERR22                                                       00769 7 14 01004 1&-4
01 041 TPERR7      LOD     14   #0020#                                                        00774 8 14 01136 1AL6
01 042             ADM     14   TPERR13          TO RETURN ADDR                               00779 6 14 00899 0HR9
01 043             TR           TPERR12                                                       00784 1    00869 0869

01 044 TPERR8      RCV     14   TPERR26   &1     CHANGE TR TO STOP 0566                       00789 U 14 00906 0I-6
01 045             TMT     14   TPERR K1  &1                                                 00794 9 14 00826 0HK6
01 046             TR           TPERR5                                                        00799 1    00739 0739

01 047 TPERR9      RCV     14   TPERR18   &1     CHANGE TR TO STOP 0901 & 0902                00804 U 14 00961 0I01
01 048             TMT     14   TPERR K2  &1                                                 00809 9 14 00831 0HL1
01 049             RCV     14   TPERR22   &1                                                 00814 U 14 01001 1&-1
01 050             TMT     14   TPERR K3  &1                                                 00819 9 14 00836 0HL6
01 051             TR           TPERR7           TO CHECK FOR ERRORS                          00824 1    00774 0774

01 052 TPERR K1    NOP          TPERR14                                                       00829 A    00914 0914
01 053 TPERR K2    NOP          TPERR19                                                       00834 A    00969 0969
01 054 TPERR K3    NOP          TPERR23                                                       00839 A    01009 1009
01 055 TPERR10     SEL                           TAPE UNIT NUMBER                             00844 2
01 056             BSP                                                                        00849 3    00004 0004
01 057 TPERR10.1   NOP          @18              BYPASS TRC                                   00854 A    00018 0018
```

FIGURE 30B. 705 LISTING FOR SERIAL TABLE SEARCH

```
PG LINE   TAG        OP      NUM    OPERAND          PROGRAM TBL SH COMMENTS          LOC OP ASU ADDRESS
01 058 TPERR11     RD                               REREAD OR REWRITE RECORD         00859 Y
01 059             TRA             TPERR12                                           00864 I     00869 0869
01 060 TPERR12     SEL             901                                               00869 2     00901 0901
01 061             TRS             TPERR16          REDUNDANCY IN MEMORY OUTPUT AREA 00874 O     00944 0944
01 062             SEL             902                                               00879 2     00902 0902
01 063             TRS             TPERR20          READING OR WRITING ERROR         00884 O     00979 0979
01 064 TPERR24     SEL                              TAPE UNIT NUMBER                 00889 2
01 065             TRS             TPERR25          END OF FILE                      00894 O     00904 0904
01 066 TPERR13     TR                               RETURN TO MAIN PROGRAM           00899 1

01 067 TPERR25     IOF                              TURN OFF INPUT/OUTPUT INDICATOR  00904 3     00000 0000
01 068 TPERR26     TR                               TO EOF ADDR OR STOP 0566         00909 1

01 069 TPERR14     SEL             500              WR EOF ERROR MSG                 00914 2     00500 0500
01 070             WR              #EOF X XXXX□#                                     00919 R     01084 1084
01 071 TPERR15     LOD     14      TPERR15                                           00924 8 14  00924 01K4
01 072             TRA             XOFF             TURN OFF 901 & 902               00929 I     01019 1019
01 073             HLT             566              END OF FILE ERROR                00934 J     00566 0566
01 074             TR              TPERR10          TO REREAD OR REWRITE             00939 1     00844 0844

01 075 TPERR16     SEL             500              WR ERROR MSG                     00944 2     00500 0500
01 076             WR              #901 X XXXX□#                                     00949 R     01095 1095
01 077 TPERR17     LOD     14      TPERR17                                           00954 8 14  00954 01N4
01 078             TR              XOFF             TURN OFF 0902                    00959 1     01019 1019

01 079 TPERR18     TR                               TO RESTART ADDR OR STOP 0901     00964 1

01 080 TPERR19     HLT             901              REDUNDANCY IN MEMORY OUTPUT AREA 00969 J     00901 0901
01 081             TR              TPERR10          TO WR AGAIN                      00974 1     00844 0844

01 082 TPERR20     SEL             500              WR ERROR MSG                     00979 2     00500 0500
01 083             WR              #902 X XXXX□#                                     00984 R     01106 1106
01 084 TPERR21     LOD     14      TPERR21                                           00989 8 14  00989 01Q9
01 085             TRA             XOFF                                              00994 I     01019 1019
01 086             NTR     15      TPERR10          TRY TO REREAD OR REWRITE RECORD  00999 X 15  00844 0HD4
01 087 TPERR22     TR                               TO RESTART ADDR OR STOP 0902     01004 1

01 088 TPERR23     HLT             902              TRIED TO RD OR WR 4 TIMES        01009 J     00902 0902
01 089             TR              TPERR10          TO TRY AGAIN                     01014 1     00844 0844

02 001 XOFF        UNL     14      XOFF3            TYPEWRITER INDICATOR OFF SUBROUTINE 01019 7 14 01054 16N4
02 002             LOD     14      #0010#                                            01024 8 14  01124 1AK4
02 003             ADM     14      XOFF3            TO RETURN ADDR                   01029 6 14  01054 16N4
02 004             SEL             901                                               01034 2     00901 0901
02 005             TRS             XOFF2            TURN OFF 0901                    01039 O     01044 1044
02 006 XOFF2       SEL             902                                               01044 2     00902 0902
02 007             TRS             XOFF3            TURN OFF 0902                    01049 O     01054 1054
02 008 XOFF3       TR                               RETURN TO MAIN PROGRAM           01054 1

       SIGNED LITERAL          1                                                     01055
       SIGNED LITERAL          4       000G                                          01059
       SIGNED LITERAL          5       0000&                                         01064

       UNSIGNED LITERAL        4       I998                                          01068
       UNSIGNED LITERAL       11       END OF JOB□                                   01079
       UNSIGNED LITERAL        4       I990                                          01083
       UNSIGNED LITERAL       11       EOF X XXXX□                                   01094
       UNSIGNED LITERAL       11       901 X XXXX□                                   01105
       UNSIGNED LITERAL       11       902 X XXXX□                                   01116
       UNSIGNED LITERAL        4       I9Z1                                          01120
       UNSIGNED LITERAL        4       0010                                          01124
       UNSIGNED LITERAL        4                                                     01128
       UNSIGNED LITERAL        4       0015                                          01132
       UNSIGNED LITERAL        4       0020                                          01136
```

FIGURE 30C. 705 LISTING FOR SERIAL TABLE SEARCH

PAGE

PROGRAM SERIAL TABLE SEARCH DATE Nov. 15, 19 IDENT. TBL SH

LINE	TAG	OPERATION	NUM	OPERAND	COMMENTS
010	READbRCD	RDTP	2	RCDbAREA ◇ EOJbRT ◇◇	
020		RAD	4	(+0007)	to step compare address
030		LRL14		LASTbCODE + 7	
040		MOVEI		BASICbADDR ◇ TBLbSEARCH ◇	
050		RAD		ITEMbCODE	
060	TBLbSEARCH	CMP			* ASU04
070		TRE		ITEMbFOUND	
080		CMP	14	TBLbSEARCH	Is this last item
090		TRE		NObCODE	Yes
100		TR		TBLbSEARCH	Continue table search
110	ITEMbFOUND	MOVEI		TBLbSEARCH ◇ COMPUTE ◇	
120		LOD	14	(1998)	to decrement by 2
130		ADM	14	COMPUTE	
140	COMPUTE	RAD			Unit cost
150		ST		UNITbCOST	
160		MPY		ITEMbQTY.	
170		RND		1	
180		ST		TOTALbCOST	
190		WRTP	1	RCDbAREA ◇◇◇	
200		TR		READbRCD	
210	EOJbRT	WRTM	1		
220		RWD			
230		TYPE		(ENDbOFbJOB ‡)	
240		HLT		9999	
250	NObCODE	HLT		1	
260		TR		NObCODE	

PROGRAM SERIAL TABLE SEARCH DATE Nov. 15, 19 IDENT. TBL SH

LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
010		DRCD			
020	RCDbAREA		1		
030	ITEMbCODE		2	+	
040	UNITbCOST		4	+	
050	ITEMbQTY.		5	+	
060	TOTALbCOST		8	+	
070		DCON			
080			1	‡	
090	FIRSTbCODE		2	03	TABLE
100			1	b	
110			4	0478‡	UNIT COST FIELD
120			49	02b1245‡01b1127‡17b0741‡	05b2993‡06b1511‡12b0387‡08b2939‡
130			49	09b3003‡19b2903‡11b2819‡	07b1584‡13b3661‡20b1054‡15b1022‡
140			28	04b0027‡18b0126‡16b1133‡	10b0247‡
150	LASTbCODE		2	14	
160			5	b1064‡	
170	BASICbADDR	RACON		FIRSTbCODE	
180					

FIGURE 31. SERIAL TABLE SEARCH

LIBRARIAN

THE LIBRARIAN is the part of the autocoder system which creates a new system tape containing a revised library. The insertion of new macro-instructions or subroutines and the replacement or deletion of some already in the library are controlled by the librarian. Input from the card reader is collated with the current autocoder system tapes to produce the new system tapes. The library on tape is organized alphabetically by macro-instruction and subroutine name. The macro-instructions precede the subroutines. Input from the card reader must be in the same sequence.

Each new addition to the library is preceded by a header card with the word INSER in the operation field. If the new entry is to replace one with the same operation code or label, the word REPLA should appear in the operation field of the header card. If a routine is to be removed from the library, only a header card need be supplied with the word DELET in the operation field and the subroutine identification in the identification field.

The following is a summary of card preparation requirements and operating instructions required for the use of the librarian.

HEADER-CARD KEY PUNCHING

A HEADER card precedes each macro-instruction or subroutine.

COLUMNS	AUTOCODER CARD FIELD	LIBRARIAN USE
1-5	Page and line	Blank
6-15	Tag	Blank
16-20	Operation	DELET, INSER, or REPLA
21-22	Numerical	No effect
23-26	Operand	Number of instructions in macro or subroutine. Left justified. (Not pertinent to DELET.)
27-74		Blank
75		M or S (depending on whether card refers to a macro or a subroutine).
76-80		Name of macro or subroutine.

The punching requirements listed below are necessary to the functioning of the librarian.

1. Each card of every macro-instruction (or subroutine) must have an M (or S) in column 75 and the name in columns 76-80.
2. Each card of a macro-instruction and subroutine including the header must be in collating sequence with respect to columns 1-5.
3. All header cards, macro-instructions and subroutines must follow in collating sequence with respect to columns 75-80.

USE OF THE LIBRARIAN

THE FOLLOWING procedure applies when the library is to be revised.

1. Place the autocoder system tape on tape unit 0207.
2. Set 0913 on.
3. Ready the deck of macro-instructions and subroutines with their header cards in the card reader.
4. Manually execute: SEL (2) 0207, RD (Y) 0000; press RESET and START.
5. The autocoder system, complete with a new list of macro-instructions and subroutines, is created on tape unit 0200 at HLT 9999.

To use the new autocoder tape, set the alteration switches for autocoder operation and push the start key.

IBM

Form 22-6726-1 (2-57; 10M—ME)