File Number GENL-22
Form C24-3259-3

IBM Systems Reference Library

Autocoder (on Disk) Program Specifications and Operating Procedures IBM 1401, 1440, and 1460

Program Number 1401-AU-008

This reference publication contains the program specifications and operating procedures for the Autocoder (on Disk) Programming System.

The specifications describe the two programs, System Control and Autocoder Assembler, that make up the Autocoder System. Logical files defined and used by the System, control cards, and results of processing operations are also included.

The operating procedures is divided into two sections. The first section describes assembling and executing object programs, changing logical-file assignments, maintaining an Autocoder library, and revising an object program. The second section describes building and updating an Autocoder System.

A summary of control card formats, phase descriptions, and a listing of a sample program make up the appendix of this publication.

For a list of other publications and abstracts, see the *IBM Bibliography* for the associated data processing system.

Preface

This publication contains the program specifications and operating procedures for the Autocoder (on disk) programming system for IBM 1401, 1440, and 1460. In this publication, the term Autocoder System or System refers to *1401/1440/1460 Autocoder (on Disk)*, program number 1401-AU-008. The language specifications for the Autocoder System are contained in the Systems Reference Library publication *Autocoder (on Disk) Language Specifications for IBM 1401, 1440, and 1460,* Form C24-3258.

This publication is divided into two major sections: program specifications and operating procedures. The program specifications describe the Autocoder System. Included in the section are such topics as a description of the System Control Program (the controlling element of the Autocoder System), a description of the processors in the Autocoder Assembler program, and a detailed description of the results of System operations. Although this section is directed primarily toward the programmer, the machine operator should review the section for an understanding of the System.

The second section, operating procedures, contains such topics as preparing processor jobs, changing file assignments for processor jobs, and running processor jobs. The last part of the section outlines the procedures to follow in building an Autocoder System. For the convenience of both programmer and machine operator, all control cards are summarized in Appendix I.

Although the second section is directed primarily to the machine operator, it is recommended that the programmer review the content of the complete section. The programmer should particularly note the parts of the section dealing with preparing processor jobs and changing file assignments.

Related Information

The following Systems Reference Library publications contain additional information relating to the use of the Autocoder System. It is recommended that these publications be available to the user for reference.

Autocoder (on Disk) Language Specifications for IBM 1401, 1440, and 1460, Form C24-3258.

Disk Utility Programs Specifications for IBM 1401, 1440, and 1460 (with 1301 and 1311), Form C24-1484.

Disk Utility Programs Operating Procedures for IBM 1401 and 1460 (with 1301 and 1311), Form C24-3105, or *Disk Utility Programs Operating Procedures for IBM 1440 (with 1301 and 1311),* Form C24-3121.

Fourth Edition

This is a reprint of C24-3259-2 incorporating changes released in the following Technical Newsletter:

Form No.	Pages Affected	Date
N21-5004	Contents, 7, 8, 8A, 11, 12, 15, 22, 23, 24, 25, 26, 27, 28, 29, 31, 43, 49, 50, 51, 52, 64	April 4, 1966

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Dept. 425, Rochester, Minn. 55901.

Contents

The Autocoder Assembler Program is one part of a language processing system that is under control of the System Control Program. (A second language processor, COBOL, is also controlled by the System Control Program.)

The Autocoder System translates source program statements written in the Autocoder language into machine-language instructions. In addition to this translating function, the Autocoder System provides these additional features:

Autocoder Library Compression. The statements that make up the library routines are compressed and stored as variable length records. This Autocoder capability ensures the efficient use of disk storage.

Relocating the Autocoder Library. The user is provided with expansion capabilities of the Autocoder library previously not possible with an Autocoder processor. Further, should the user wish, he can relocate the Autocoder library to an area of his choice in disk storage.

Building Multiple Autocoder Libraries. In addition to being able to relocate the Autocoder library, the user can also build more than one Autocoder library. Small libraries that contain selected routines appropriate to particular types of job processing significantly reduce library-change time.

Changing Input/Output Devices. The Autocoder System provides the user with the option of changing the form of input to and output from specific jobs. So that the Autocoder System can operate at a machine-independent level, a set of logical files that are used for input/output operations has been defined. Although these logical files are assumed by the System Control Program to be assigned to a defined set of input/output devices, the user can change these assumptions according to his particular needs.

Stacking of Jobs. Under control of the System Control Program, it is possible to process a series of jobs without regard to the type of processing that is being performed. For example, it is possible to assemble source program number one, partially assemble source program number two, and execute object program number three, all in one stack.

Building an Object-Program Library in Disk Storage. By using one of the logical files (CORELOAD) defined by the Autocoder System, it is possible to build an object-program library in disk storage. Because the upper and lower limits of each object program stored in this area in disk storage are supplied to the user by the Autocoder System, the user has immediate access to any one of the stored object programs. Using an object-program library substantially reduces program load time (as opposed to loading from cards) and eliminates excessive handling of punched-card object decks.

Executing Punched-Card Object Programs. If a program is infrequently used, the user may wish to maintain a punched-card object program, thus saving disk storage for other purposes. When this is the case, the user has two options for executing this object program. It can be executed either under control of the Autocoder System (as a job in a stack of jobs), or it can be executed completely independent of the System.

Definition of Key Terms

To clarify the meaning of special terms used in this publication, the following definitions are given. Standard terms are defined in *Glossary for Information Processing*, Form C20-8089.

Assembler. The program that translates Autocoder symbolic statements into actual machine language. This process is called an *assembly.*

Autocoder Text. A series of 100-character records containing the source-program statement, or a generated statement, and assembly information.

Batched Files. Logical files whose contents represent one or more sequential sets of input to or output from the Autocoder System.

Bootback. A routine located in upper core storage during execution that provides linkage between the user's object program and the System Control Program. This linkage is required when executing an object program in a stack of jobs.

Card Boot. A card deck, supplied as part of the Autocoder System program deck, that is used to start all System operations.

Job. An operation or series of operations to be performed by the Autocoder System.

Logical Files. Input/output devices and/or areas used by the Autocoder System.

Object-time. A term describing those elements or processes related to the execution of a machine-language object program.

Operation. A basic unit of work to be performed by one of the components of the System.

Stack. A set of one or more jobs to be processed during the same machine run.

System. The set of programs made up of the elements required for assembling and/or executing user-programs.

[ ] Brackets contain an option that may be chosen.
{ } Braces contain options, one of which must be chosen.

Machine Requirements

The Autocoder System requires the following minimum machine configurations.

An IBM 1401 system with:
- 4,000 positions of core storage
- High-Low-Equal Compare Feature
- One IBM 1311 Disk Storage Drive
- One IBM 1402 Card Read-Punch
- One IBM 1403 Printer.

An IBM 1440 system with:
- 4,000 positions of core storage
- One IBM 1301 Disk Storage or one IBM 1311 Disk Storage Drive
- One IBM 1442 Card Reader
- One IBM 1443 Printer.

An IBM 1460 system with:
- 8,000 positions of core storage
- One IBM 1301 Disk Storage or one IBM 1311 Disk Storage Drive
- One IBM 1402 Card Read-Punch
- One IBM 1403 Printer.

The Autocoder System can use the following devices and features if available:
- IBM 1404 Printer
- IBM 1444 Card Punch
- Console Printer
- 8,000, 12,000, or 16,000 positions of core storage
- Print Storage feature
- Direct Seek feature (for a library change only).

The Autocoder System

The Autocoder System built by the user contains the System Control Program and the Autocoder Assembler Program.

System Control Program. The System Control Program is the controlling element of the System. Its main function is to analyze control-card information, and transfer control to the appropriate portion of the system.

Autocoder Assembler Program. The Autocoder Assembler Program translates source programs, written in the Autocoder language, into machine-language object programs. The object programs can subsequently be executed by the Autocoder System.

System Control Program

All system operations are initiated by a deck of cards supplied by IBM. This deck, called the Card Boot, reads in the first portion of the System Control Program from disk storage. Ultimately, the entire resident portion of the System Control Program is read into lower core storage.

All control-type functions for the System are accomplished by the System Control Program. These functions include:

Assigning Input/Output Devices. Input/output operations are coordinated with user-specified input/output devices.

Updating the System. The System Control Program updates the system to the latest modification level or version.

Selecting Appropriate Processor Runs. Through control cards supplied by the user, the System Control Program determines the operations necessary for the completion of a job. For example, if a source program is coded in the Autocoder language, and the end result of processing is to be a machine-language object program, processing must be performed by the Autocoder processor and the Output processor. The control card says in effect that the source program is coded in Autocoder and that processing is to run through the Output processor. The System Control Program reads the control card and calls in the Autocoder processor. Processing takes place, and at the completion, control reverts to the System Control Program. The System Control Program then calls in the Output processor. Processing takes place, and at the completion, control again reverts to the System Control Program. Because the Output processor was the last processor to be selected, the System Control Program reads the control card for the next job.

The remainder of this section describes the following aspects of the System Control Program:
- Logical Files
- Control Cards.

Logical Files

A set of logical files, defined by the Autocoder System, is used for input/output operations. Each file has a specific function and is assigned by the System Control Program to a particular device. The user can alter

the file-assignments by using ASGN (assign) control cards. (See *Changing File Assignments.*)

The logical files may be thought of as falling into one of four general categories. These categories are:

Residence File
Operation Files
External Files
Internal Files.

The functions of the logical files and the devices to which they can be assigned follow.

Residence File

SYSTEM File. The SYSTEM file contains the System Control Program and the Autocoder Assembler Program. It is assigned to a fixed area in a 1311 or 1301 disk unit.

Operation Files

CONTROL File. The CONTROL file contains cards or card images that send control information to the System Control Program. It can be assigned to the card reader or the console printer.

MESSAGE File. The MESSAGE file contains information of primary interest to the machine operator. These messages are usually diagnostics relating to the operating procedures and/or instructions to the machine operator. It can be assigned to the printer or the console printer.

External Files

LIST File. The LIST file, generally associated with high-volume printed listings, contains information directed primarily to the source programmer. It can be assigned to the printer, or to disk storage, or it can be omitted. If the LIST file is assigned to a disk unit, the information is stored two sectors per printed line in the move mode.

INPUT File. The INPUT file contains source information to the processors. It can be assigned to the card reader or to any available area in disk storage. If the file is assigned to a disk unit, the card images must be stored one card per sector in the move mode.

OUTPUT File. The OUTPUT file contains the results of the operation or series of operations specified in the RUN card. It can be assigned to the card punch, or to disk storage, or it can be omitted. If the file is assigned to a disk unit, any card images will be stored one per sector in the move mode.

LIBRARY File. The LIBRARY file is a disk-storage file that supports the Autocoder macro facility. This file contains the library table and library routines, such as IOCS. It is maintained by the Autocoder Librarian and used by the Autocoder Macro Generator. The LIBRARY file can be assigned to any available area in disk storage.

CORELOAD File. The CORELOAD file is a disk-storage file used by the Output and Execution processors of the Autocoder Assembler Program. The file contains an object program in the load mode. The CORELOAD file is developed by the Output processor and is used by the Execution processor.

Note. Only the external files INPUT, OUTPUT, CORELOAD, and LIST can be batched. Batching will be performed when a series of jobs is processed without intermediate file assignments to these external files. When batch processing is performed, input to and output from the processors is stored sequentially within the files.

Internal Files

WORK1 File. The WORK1 file contains the intermediate results from the Autocoder processor. It can be assigned to any available area in disk storage.

WORK2 File. The WORK2 file is used by the Autocoder processor. It contains information for the cross-reference listing and can be assigned to any available area in disk storage.

WORK3 File. The WORK3 file is used by the Macro Generator, the Autocoder processor, and the Output processor. It can be assigned to any available area in disk storage.

Control Cards

The System Control Program recognizes seven types of control cards. They are:

RUN
ASGN
INIT
UPDAT
NOTE
PAUSE
HALT

Each type is punched in the Autocoder format. Appendix I contains a summary of all specific control cards that the System Control Program recognizes. Included in Appendix I is a detailed description of the manner of punching each specific control card and valid entries for each of the general formats as discussed in the following sections.

RUN Card

The RUN card indicates the portion(s) of the Autocoder Assembler Program that are to be selected by the Sys-

tem Control Program. A RUN card is required for each job to be performed. The general format of the RUN card is:

$$\left\{\begin{matrix}\text{AUTOCODER}\\ \text{OUTPUT}\\ \text{EXECUTION}\end{matrix}\right\} \text{RUN} \left[\text{THRU} \left\{\begin{matrix}\text{OUTPUT}\\ \text{EXECUTION}\end{matrix}\right\}\right]$$

If the optional part of the RUN card is omitted (THRU OUTPUT OF THRU EXECUTION), the System Control Program assumes that only the named processor is to be selected. The THRU option enables the System Control Program to call a series of processors automatically.

Valid entries for the RUN card are:

AUTOCODER RUN
AUTOCODER RUN THRU OUTPUT
AUTOCODER RUN THRU EXECUTION
OUTPUT RUN
OUTPUT RUN THRU EXECUTION
EXECUTION RUN

See *Preparing Jobs* for the specific RUN card format required for each job.

ASGN Cards

An ASGN card indicates to the System Control Program that a logical file is to be assigned to a specific input/output device. An ASGN card is used when the user wants a logical file assigned to an input/output device or area other than the assumed assignment of the System Control Program, or when the user wants to change an assignment that he has previously made.

The general format for an ASGN card is:

$$\textit{file-name} \quad \text{ASGN} \quad \left\{\begin{matrix}\textit{device}\\ \text{OMIT}\end{matrix}\right\}$$

The *file-name* is the specific logical file; *device* is the input/output unit to which the logical file is to be assigned. Two examples for using an ASGN card follow.

The logical file, INPUT, is to be changed from the assumed device assignment (READER 1) of the System Control Program to an area in disk storage. This area is to be on 1311 unit 3, beginning at address 000600 and extending *to* (not through) 000900. Note that the END address to be punched is one more than the area actually used by the INPUT file. The ASGN card for this example is punched:

```
INPUT ASGN 1311 UNIT 3, START 000600, END 000900
```

The second example illustrates the omission of a logical file. (This option is valid only in specific cases.) If the OUTPUT file is to be omitted, the ASGN card is punched:

```
OUTPUT    ASGN    OMIT
```

The user must leave blanks between items in the operand field where indicated in the specific formats. For example, if the operand is READER 2, there must be a blank between READER and 2.

During a single stack of jobs, an assignment made by the user for a single logical file remains in effect until a HALT card, an INIT card, or another ASGN card is sensed for that particular file. For example, an ASGN card that specifies the INPUT file to be assigned to READER2 causes the assumed assignment, READER1 to be altered. The System Control Program will select READER2 during a single stack until an INIT card or another ASGN card for the INPUT file is encountered.

INIT Card

The INIT card indicates to the System Control Program that all assumed logical file assignments are to become effective. The general format of the INIT card is:

```
INIT    any message
```

An INIT card can occasionally be used as a convenient substitute for an ASGN card. For example, assume that the INPUT file is assigned to disk for a particular job in a stack. If the next job is to be read in from READER1, the INPUT file assignment must be changed from disk to READER1. For this purpose, an INIT card may be used instead of an ASGN card because READER1 is the assumed assignment for the INPUT file.

UPDAT Card

The UPDAT card is included in a package supplied by IBM for the purpose of updating the user's Autocoder System. It is prepunched in the following format:

$$\left\{\begin{matrix}\textit{processor-name}\\ \text{SYSTEM}\end{matrix}\right\} \text{UPDAT} \quad \textit{phase-name,} \left\{\begin{matrix}\text{ALL}\\ \text{DELETE}\\ \text{HEADER}\\ \text{INSERT}\\ \text{PATCH}\end{matrix}\right\}$$

This card (excluding DELETE) will be followed by the appropriate data cards.

NOTE Card

The NOTE card contains messages and/or instructions from the programmer to the machine operator. Processing is not interrupted when the System Control Program senses this control card. The contents of the NOTE card are printed on the MESSAGE file. The general format of the NOTE card is:

```
NOTE    any message and/or instruction
```

A NOTE card could be used when the programmer wants to direct that the output from a series of conven-

tional assemblies be placed on the CORELOAD file located on disk drive 2. At the completion of processing the series of jobs, a NOTE card could be used to tell the machine operator to remove the disk pack from drive 2. The message would be:

```
NOTE      REMOVE DISK PACK FROM DISK DRIVE 2
```

PAUSE Card

The PAUSE card contains messages and/or instructions from the programmer to the machine operator. When the PAUSE card is sensed, the System Control Program

temporarily halts the system. The contents of the PAUSE card are printed on the MESSAGE file. Processing is resumed by pressing the start key. The general format for the PAUSE card is:

PAUSE *any message and/or instruction*

One application of the use of a PAUSE card might be in the case where the INPUT file for a job is located on disk unit 3. The programmer could inform the machine operator of this fact by using a PAUSE card, telling him to ready the drive. The message would be:

PAUSE READY THE PACK ON DISK DRIVE 3

HALT Card

The HALT card indicates to the System Control Program that processing has been completed. It is the last card of a stack. The contents of the HALT card are printed on the MESSAGE file. The general format for the HALT card is:

HALT *any message and/or identification*

Autocoder Assembler Program

The Autocoder Assembler Program is made up of the following sections:

Preprocessor
Autocoder Processor
Output Processor
Execution Processor.

Preprocessor

The Preprocessor consists of four portions, each of which has a specific function:

Option Control. The Option Control analyzes control card information and determines the operation(s) to be performed. It then transfers control to the Librarian, Update, or Macro Generator.

Librarian. The Librarian maintains the Autocoder library by inserting, deleting, and/or modifying the library routines according to the user's specifications. Whenever the contents of the library are changed, the Librarian updates the library table which is the directory of library routines.

Update. The Update portion performs the function of updating all portions of the Preprocessor.

Macro Generator. The Macro Generator performs preassembly operations. It analyzes the Autocoder source program to determine if it includes any macro instructions. For each macro named in the source program, the Macro Generator extracts the associated routine from the library, tailors the routine if parameters are supplied in the macro instruction, and generates a routine in the Autocoder format.

Two of the three Preprocessor portions (Librarian and Update) that are called by the Option Control complete the job requested by the user. The results of the Librarian operations can be an updated library, a listing of the library table, and/or a listing of routines. An Update operation causes the Preprocessor to be updated to the latest version or modification level of the Autocoder System. At the successful completion of each of these operations, control returns to the System Control Program.

The Macro Generator performs only the first step in a program assembly. The result of the Macro Generator operation is an Autocoder source program that contains tailored library statements. The next step, translating source statements into machine language, is performed by the Autocoder processor.

Autocoder Processor

The Autocoder processor diagnoses the source statements and converts the symbolic references in the source statements to actual machine codes and addresses. The processor arranges the results of its operations to produce Autocoder text.

Autocoder text is a series of 100-character records. Each record contains a source-program statement, or a generated statement, and assembly information such as the machine-language instruction, the length and address of the instruction, and diagnostic flag symbols.

The results of Autocoder processing and the operations required to produce the results are:

Operation	*Result*
Diagnose source statements	Diagnostic messages and flag symbols
Convert symbolic to actual	Label table and flag symbols
Arrange results of assembly	Autocoder text (100-character records)

At the completion of Autocoder processing, the text is ready for the Output processor, which develops various forms of output.

Output Processor

The Output processor rearranges the Autocoder text according to the user's specifications.

The results of Output processing and the rearrangement required to produce the results are:

Result	*Rearrangement*
Program listing	The text is edited. Blanks are inserted between items of information. Headings to identify the items are incorporated in the listing. A sequence number is assigned to each statement on the listing.
Resequenced source deck	Source statements are extracted from the text, and sequence numbers are substituted for page and line numbers.
Object program (card format)	Machine-language instructions are extracted from the text, and the necessary loading instructions are incorporated.
Object program (coreload format)	Machine-language instructions are extracted from the text and transferred to disk storage.

Object programs, in either format, are ready to be executed. Execution of object programs in the coreload format must be handled by the Execution processor. Execution of object programs in the punched-card format can be handled by the Execution processor or executed independent of the System.

Execution Processor

The Execution processor starts execution of the object program and provides linkage with the System Control Program so that the next job can be performed, without operator intervention, immediately after execution of the object program.

The Processor reads the bootback routine (linkage) into upper core storage, calls the object program, and transfers control to the object program.

Linkage to the bootback routine can be established by using the SYSCL macro or by a manual branch to the routine.

As described under *Output Processor,* the object program can be in card format, which includes loading instructions, or in coreload format, which requires a disk loader. The Execution processor supplies the disk loader required by an object program in coreload format.

Thus, the Execution processor permits the user to include his object programs within a stack of jobs to be performed.

Results of Processing Operations

The results of processing operations can be divided into the following categories:

1. Documentation. Control card diagnostics, source statement diagnostics, label tables, cross-reference listings, and program listings fall into this category.
2. Intermediate results in the development of an object program (Autocoder text).
3. Object programs in card or coreload format.
4. Messages that specify the disk storage location of any results that are to be used for future processing.
5. Resequenced source deck.
6. Execution of object programs. Execution of object programs can be accomplished under control of the System Control Program, or independent of the System. See *Using and Maintaining the Object Program.*

Documentation

Control Card Diagnostics

If any invalid characters are detected in the CTL card (control card for assembly), the CTL card image and the diagnostic message(s) are listed. The messages inform the user that his CTL card is invalid. The halt gives the user the opportunity to decide if the assembly should be continued.

The CTL diagnostic messages and the format of the CTL card are shown in Figure 1.

Source Statement Diagnostics

The Autocoder processor phases, which analyze source statements and develop diagnostic messages, are optional. Their inclusion or exclusion is specified in the source-program CTL card.

If any errors are detected in source-program statements during the diagnostic phases, the invalid statements (except columns 13-15 and 73-80) are listed. A message appears at the right of each invalid statement. If the statement contains more than one error, the diagnostic message refers to the first error detected. The halt that occurs after the diagnostic phases have been completed gives the user an opportunity to decide if the assembly should be continued.

If the errors are not corrected, flag symbols may appear on the program listing and the object program, when executed, may not produce the intended results (see Figure 2).

Label Table

The label table lists all labels and their equivalent addresses. Area-defining literals, followed by the # sign, are also included. The labels and area-defining literals are listed in alphabetical order according to the first character. Indexing is indicated as shown in the sample label table (Figure 3).

The maximum number of labels and area-defining literals that can appear in the label table depends on the number of disk-storage sectors assigned to the WORK3 file. See *File Considerations* under *Changing File Assignments.*

Any errors detected by the Autocoder processor are indicated by the following flag symbols:

A Name equated to an area-defining literal.

Diagnostic Messages	CTL Card Format		
Card Image of Invalid CTL Card	Column	Indicates	Contents
	16-19	Mnemonic	CTL
INVALID MACHINE SIZE SPECIFIED, 4K ASSUMED	21	Object-machine size	1 (4K); 2 (8K); 3 (12K); 4 (16K)
INVALID CHAR COL 22, BLANK ASSUMED	22	Modify address	1 (yes); not punched (no, if the object machine is 4K; or yes, if the object machine is 8K, 12K, or 16K)
INVALID CHAR COL 23, BLANK ASSUMED	23	Advanced programming or index and store-address register feature.	1 (yes); not punched (no)
INVALID CHAR COL 24, BLANK ASSUMED	24	Multiply-divide feature	1 (yes); not punched (no)
INVALID MACHINE SPECIFIED, PROCESSOR MACHINE ASSUMED	25	Object machine	0 (1401); 4 (1440); 6 (1460)
INVALID CHAR COL 26, x ASSUMED (x = P for 1401 and 1460; x = S for 1440)†	26	Punch device	S (1442 or 1444); P (1402)
INVALID CHAR COL 27, x ASSUMED (x = P for 1401 and 1460; x = S for 1440)†	27	Read device	S (1442); P (1402)
INVALID CHAR COL 28, x ASSUMED (x = P for 1401 and 1460; x = S for 1440)†	28	Printer device *	S (1443); P (1403)
INVALID CHAR COL 29, 1 ASSUMED	29	Disk device	1 (1311 or 1301); 2 (1405)
INVALID CHAR COL 30, BLANK ASSUMED	30	Source statement diagnostic	N (no); 1 or not punched (yes)
INVALID CHAR COL 31, BLANK ASSUMED	31	Label table or cross-reference listing.	L (Label Table); N (Neither); not punched (cross-reference listing
INVALID READ-IN LOCATION, 00001 ASSUMED	32-36	a. Object program in self-loading format b. Read-in area for a 1440 object program in the condensed-loader format.	a. Sbbbb (object program in self-loading format) b. 5-digit starting address, or not punched (starting address of the 1440 read-in area is 00001) Note: Leave blank for a 1401 or 1460 object program in the condensed-loader format.
INVALID LOADER LOCATION, 000xx ASSUMED (xx = 81 for 1401 and 1460, xx = 75 for 1440)	37-41	Loader location	5-digit starting address. If column 42 contains a D, punch: 03701 for 4K 11701 for 12K 07701 for 8K 15701 for 16K These columns are not checked if column 32 contains an S
INVALID CHAR COL 42, BLANK ASSUMED	42	Disk loader(for object programs in the coreload format)	D (yes); not punched (no)

† The values of x depend on the object machine specified in column 25.
* Consider a 1403 Printer attached to a 1440 system as being the same as a 1443 Printer.

Figure 1. CTL Diagnostics and CTL Card Format

M Multiply defined. The same label appears in more than one label field.

E Invalid operand in an EQU statement.

Cross-Reference Listing

The cross-reference listing lists all labels and area-defining literals used in the program. The address assigned to the label or literal and the sequence numbers of the statements in which the label or literal is used are given. For a label, the first sequence number listed is the sequence number of the statement that defines the label; for an area-defining literal, the first sequence number listed is the sequence number of the first statement that uses the literal.

The maximum number of labels and area-defining literals that can appear in the cross-reference listing depends on the number of sectors assigned to the WORK3 file. The maximum number of references to labels and area-defining literals depends on the number of sectors assigned to the WORK2 file. *See File Considerations* under *Changing File Assignments.*

The labels and area-defining literals are listed in alphabetical order. Each literal is followed by a # sign in the tag column. If a label is undefined, it appears with all sequence numbers assigned to it and with periods in the address column. A zone bit over the tens position of the address indicates that the label is indexed. The zone bit used is the same as that which appears in the machine language address.

Diagnostic Message	Meaning	Processor Action
OPERATION	The operation field does not contain a valid mnemonic or a machine-language operation code.	1. An eight-character no-operation instruction (N xxx xxx x) is inserted. 2. If an operand or the d-character is not specified, the assembler inserts zeros.
F FORMAT	An operand is invalid: 1. An operand contains one of the following special characters , # + - b . 2. Invalid literal. 3. Literal used in an EQU, ORG, or LTORG statement. 4. Blank operand used in a declarative or EQU statement.	1. If the statement is a DA header, a subsequent DA, or a DS, the operand is replaced with 1. 2. For a DSA the count is 3. For a DCW or DC the count is: 1 for a blank constant; 50 for an alphameric constant; 3 for an address constant; equal to the number of numeric characters in a numeric constant.
L FORMAT	A symbolic operand exceeds six characters, or an actual address operand exceeds five characters.	1. Three periods replace the operand.
X FORMAT	An X-control field is invalid.	1. The invalid X-control field is processed.
D-MODIFIER	A d-modifier is missing or is invalid for the operation specified.	1. A blank is inserted if the d-modifier is missing. 2. The statement is assembled with the invalid d-modifier.
ADJUSTMENT	An indexing or adjustment factor is used incorrectly.	1. If double indexing is specified, the last index factor is used. 2. If the adjustment factor is invalid, it is ignored.
LABEL ERR	A label is invalid: 1. It exceeds six characters. 2. It begins with a numeric character, or it contains one of the following special characters , # + - b . 3. It is missing in an EQU statement.	1. Extra characters are deleted. 2. The label is processed with the special characters. If the label is used as an operand in another statement it will be recognized as an erroneous operand.
OPRND TYPE	The A- or B-operand is invalid for the operation specified. For example, %G2 is invalid in MLC NAME, %G2.	The statement is assembled with the invalid operand.
# OPERANDS	An operand is missing, or there are too many for the operation specified.	1. If an operand is missing in an I/O instruction that requires eight characters, periods are inserted; otherwise, the statement is assembled as specified. 2. Extra operands are dropped.

Figure 2. Source Statement Diagnostics

Any errors detected by the Autocoder processor are indicated by an A, M, or E in the tag column. The meanings of these symbols are given under *Label Table*. The cross-reference listing associated with the sample program (supplied with the Autocoder program deck) is shown in Appendix III.

Program Listing

The program listing documents the program and enables the programmer to see the results of Autocoder processing. The listing also assists the programmer if revising the program is necessary.

The following messages, if appropriate, appear at the end of a program listing:

END OF LISTING—X ERRORS, where X is the number of program errors.

OBJECT CORE EXCEEDED, which counts as a program error.

X or NO SEQUENCE ERRORS, which does not count as a program error.

A description of the 120-character and 100-character listings follows.

```
                                LABEL  TABLE

AREA  #  01082      AREA1 #  01085      CHECK    01037      DELAY    01062      END      01070
LABEL    01007      LABEL1   01023      LABEL2 M 01030      LABEL2 M 01041      RESULT   01022&X1
SUBTOT   01022      TOTAL  E .....      X1       00089
```

Figure 3. Label Table

120-Character Listing

Program errors are indicated by flag symbols in the last columns of the program listing. The thirteen flag symbols and their meanings are:

- # Invalid number of operands
- O Invalid operation code
- D Invalid d-modifier
- X Invalid X-control field
- F Format error
- L Extra characters in a symbolic or actual address operand
- A Invalid indexing or adjustment
- I Invalid symbolic indexing
- U Undefined operand
- E Reference to the label of an invalid EQU statement
- M Reference to a multiply defined label
- C Result of address adjustment is greater than 16,000 or less than zero
- S Source statement is out of sequence.

The format of the 120-character listing is:

Columns	*Contents*
1-4	Sequence number assigned by the Output processor
5	Blank
6-10	Source program page and line number
11	Blank
12-18	Label or blank
19	Blank
20-24	Operation code mnemonic
25	Blank
26-77	Operands and comments
78	Blank
79	Suffix character or blank
80	Blank
81-82	Count (number of characters in the assembled instruction), or blank. Blank constants and area-defining literals have no count.
83-84	Blank
85-89	Location of the assembled instruction
90-91	Blank
92-99	Assembled instruction
100	Blank
101-105	A-address (actual) or X-control field
106	Blank
107-111	B-address (actual)
112-113	Blank
114	Period
115	Label error flag
116	Operation error flag
117	A-operand flag
118	B-operand flag
119	d-modifier flag
120	Sequence flag

100-Character Listing

The format of the 100-character listing is the same as the 120-character listing except that the suffix character, the count, the location of the assembled instruction, and the assembled instruction are shifted three positions to the left. The A- and B-addresses are omitted. Column 100 contains a W flag symbol which is a warning that the statement contains an error.

Autocoder Text

The Autocoder text is a series of 100-character records that are developed by the Autocoder processor. Each record contains a source-program or generated statement and assembly information such as the machine-language instruction, the length and address of the instruction, and diagnostic flag symbols.

The Autocoder text can be used as a restart point for Output processing.

Object Programs

Card Formats

Two object-program card formats, self-loading and condensed-loader, are available. The condensed-loader card deck (Figure 4) consists of object-program cards which are preceded by clear cards, a bootstrap card, and load cards. The loader instructions for 1440 normally require 132 positions of core storage; the loader instructions for 1401 and 1460 require 125 positions. (See *Condensed Loader Considerations.*)

The self-loading card deck (Figure 5) consists of cards that contain loading instructions, and object-program instructions and/or data. Two clear-storage cards and a bootstrap card precede the self-loading cards.

A 1440 object program in the self-loading format requires that the read-in area be 00001-00072 and that positions 73-85 be reserved for the read-a-card and branch instructions, which are moved into these positions by the bootstrap routine. A 1401 or 1460 object

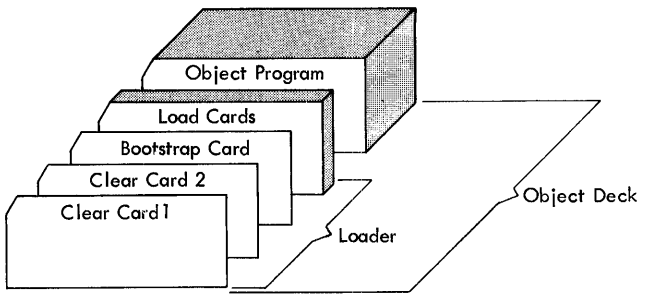

Figure 4. Object Deck in the Condensed-Loader Format

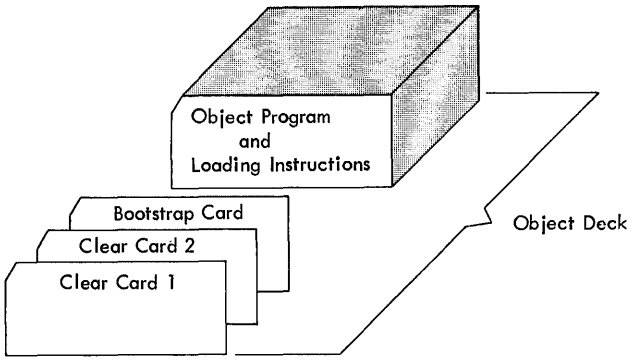

Figure 5. Object Deck in the Self-Loading Format

program requires no additional positions outside the read-in area.

An execute card in either a condensed-loader or self-loading object deck interrupts the loading, so that a portion of the object program that has already been loaded is executed. If a source program EX or XFR statement caused the execute card to be developed, the object program must contain a branch instruction that transfers control back to the loading instructions. If a DA statement caused the execute card to be developed, the execute card contains instructions that prepare the defined area according to the specifications in the DA header (clear the area, set word marks, create record marks, create a group-mark word-mark). The execute card also contains a branch back to the loading routine.

Note: Generally, on a 1442, an object deck in the condensed-loader format can be loaded faster than a deck in the self-loading format.

On a 1402, an object deck in the self-loading format can be loaded faster than a deck in the condensed-loader format.

Condensed-Loader Format

The cards that precede the object program are called the *loader* (six cards for 1401 or 1460, seven cards for 1440). The first two cards in the loader contain instructions to clear storage before the program is loaded. Columns 79-80 contains C1 in card 1 and C2 in card 2.

The third card is the bootstrap card. This card sets word marks for the instructions in the load cards and supplies an instruction that reads the load cards into the read-in area. The bootstrap card contains identification in columns 73-80. For a 1440 program, the identification is BOOTSTCD; for a 1401 or a 1460 program, it is BOOTLD01.

The remaining cards in the loader are the load cards. These cards contain the loading routine and the instructions that move the loading routine into the loading instruction area.

After the loading instruction area has been initialized, control is transferred to the loader. The loader moves the data and instructions in the object-program deck into their proper locations in core storage.

The object program cards have the following format:

Columns	*Contents*
1-3	The three-character machine address of the first storage position to be loaded.
4-5	The number of characters to be loaded from the card. Word-separator characters are not counted.
6-71	The instructions and/or constants to be loaded. A word-separator character (0-5-8 punch) precedes every character requiring a word mark in core storage. Each pair of word-separator characters is loaded as a single word-separator character with no word mark. An odd number of word-separator characters (n) is loaded as $\frac{n-1}{2}$ word-separator characters with no word marks; the last word-separator character causes a word mark to be set in the position that will contain the next character in the card.
72-75	The program-listing sequence number of the first instruction or constant to be loaded.
76-80	Identification. The identification in columns 76-80 of the JOB card appears in all cards in the condensed deck. Each new JOB card in the source deck causes the identification of the condensed deck to be changed.

Self-Loading (1401 and 1460)

The first two cards in the self-loading object deck are clear cards that clear storage before the object program is loaded. These cards are identified by C1 and C2 in columns 79-80.

The third card is a bootstrap card that contains instructions that set word marks in the read-in area before the object program is loaded. This card is identified by BOOTSTRAP in columns 72-80.

The remainder of the cards contain assembled program instructions and load instructions. There can be as many as seven instructions or constants on each card. The card format is as follows:

Columns	*Contents*
1-39	The instruction and/or constants to be loaded into core storage.
40-46	Instructions that load the instructions or constants into core storage with a high-order word mark.
47-67	Three 7-character set-word-mark instructions (or one clear-word-mark and two set-word-mark instructions for cards beginning with partial instructions or constants that do not require a high-order word mark). These instructions set the word marks that define the separate fields in the block of core storage being loaded.

Columns	*Contents*
68-71	1040. This is an instruction to read a card and branch to location 040.
72-75	Program-listing sequence number of the first instruction or constant to be loaded.
76-80	Identification. The identification in columns 76-80 of the JOB card appears in all cards in the self-loading deck. Each new JOB card in the source deck causes the identification of the self-loading deck to be changed.

Self-Loading (1440)

The first two cards clear core storage before the program is loaded. These cards are identified by C1 and C2 in columns 79-80.

A bootstrap card, identified by M%G1001R in columns 73-80, loads a group-mark word-mark, a read-a-card instruction, and a branch instruction (B040) into positions 72-84 of the read area. Position 85 must be left blank or contain a wordmark.

The format of the remaining cards is the same as that described for 1401 and 1460, except columns 68-71 contain B073. This instruction causes a branch to 073 which contains the bootstrap card read-a-card instruction.

Coreload Format

An object program in the coreload format is written in disk storage. It contains the machine-language object-program instructions. At execution time a disk loader, supplied by the Execution processor, initiates the loading of the object program.

The object program in coreload format is written in the load mode. The structure of the program in disk storage is:

1. A one-sector header record that has the following format:

Positions	*Contains*
1-7	A move instruction that transfers the address of the first operating sector to the disk loader.
8-11	A branch to the disk loader.
12-17	The address of the first operating sector.
18-23	HEADER
24-28	The identification from the last JOB card in the source program, or blank if no JOB card was included.
29-80	The operand from the last JOB card in the source program or blank if no JOB card was included.
81-90	Unused

2. Full 90-character sectors. These sectors contain an exact core-storage image of the object program.
3. Operating sectors. The first sectors contain instructions that load the full 90-character sectors into their proper core-storage locations. The remaining sectors contain instructions that fill in the instructions and/or constants that could not be put into a full 90-character sector during the Output processor operation.
4. An execution instruction that causes a branch to the object program at object-time.

If the source program contains EX or XFR statements, sections 2, 3, and 4 are repeated for each overlay.

Note: Certain restrictions must be considered when writing a source program that is to be an object program in the core-load format:

1. A group mark must not be the first character of a literal or the first data character of a DCW statement.
2. Before returning control to the disk loader for loading a new program or program overlay, any group-mark word-marks within the section of core storage being overlaid should be cleared.
3. Statements within a program or program overlay are not always loaded into core storage in the same order they were coded.

Messages

One of the following messages appears when the input for or output from an operation is assigned to disk storage.

1. { INP / OUT / LST } FILE { STARTS / ENDS } ON { 1311 / 1301 }

   AT ADDRESS *nnnnnn.*

2. CORELOAD OUTPUT COMPLETE ON { 1311 / 1301 } UNIT *n,*

   START *nnnnnn,* END *nnnnnn.*

The messages that reflect the location of results stored in disk files should be recorded because the addresses specify restart points for future processing.

Resequenced Source Deck

A resequenced source deck is the original source program with the page and line numbers (columns 1-5) replaced by sequence numbers assigned by the Output processor. The numbers start with 0001 in columns 1-4. Subsequent entries are increased by 0001. The format of the resequenced deck is:

Columns	*Contents*
1-4	Sequence number assigned by the Output processor (0001-xxxx)
5	Blank
6-72	Columns 6-72 of the source card
73-75	Blank
76-80	Identification from the JOB cards as encountered in the source deck

Jobs

The Autocoder System performs three major operations.

1. Translates source programs.
2. Produces object programs.
3. Starts the execution of object programs.

Because these operations are performed by the three processors of the System, the operations are called *processor jobs*. In this respect, the Autocoder processor translates source programs. The Output processor produces object programs. The Execution processor starts the execution of object programs.

Two other operations, maintaining the Autocoder library and updating the Autocoder System, are also considered jobs. Maintaining the Autocoder library is called a *library job*. Updating the Autocoder System is called an *update job*. Update jobs are described in *Updating an Autocoder System.*

Under control of the System Control Program, it is possible to perform one or more jobs without operator intervention. This process is called *stack processing*. A stack is *always* made up of the Card Boot deck, a SYSTEM ASGN card, the particular job(s) to be performed, and a HALT card.

In performing a job, the following must be taken into consideration.

1. The kind of input for the job.
2. The use of the logical files.
3. The machine-operator procedures to be followed.

The kinds of input for processor jobs and library jobs are discussed in the following sections *(Preparing Processing Jobs* and *Preparing Library Jobs).*

The general use of logical files is discussed in *Logical Files.* In most cases, the user does not need to be concerned about the logical files used for a particular job because the Autocoder System defines the files and assigns them to specific input/output devices. In the description that follows of preparing individual processor jobs, any file asisgnment that the user must make is explained.

The machine-operator procedures to be followed are described in *Performing Jobs.*

Preparing Processor Jobs

The kind of output that is desired by the user is the determining factor of which processor job is to be performed. Figure 6 lists each processor job and the output from the Autocoder System by the completion of the job. In the figure, YES means that the type of output is *always* produced. OPT means that the type of output is produced only if the user specifies that it be. This is done by supplying output option (OPTN) cards in addition to the required control cards.

The remainder of this section describes each individual processor job. They are:

AUTOCODER RUN THRU OUTPUT
AUTOCODER RUN THRU EXECUTION
EXECUTION RUN
AUTOCODER RUN
OUTPUT RUN
OUTPUT RUN THRU EXECUTION

Each processor job description includes:

1. Assumed input device. This entry refers to the device on which the INPUT file is assumed to be located. For the 1402, READER 1 means that the cards are selected into stacker 1. For the 1442, READER 1 means unit 1.
2. Input. This entry refers to the type of input for the job.
3. Assumed output device. This entry refers to the device on which the LIST file, the MESSAGE file, and the OUTPUT file are assumed to be located. For the 1403, PRINTER 2 means that 132 print positions are available. For the 1443, PRINTER 2 means that 144 print positions are available. For the 1402, PUNCH 4 means that the cards are selected into stacker 4. For the 1442, PUNCH 1 means unit 1.
4. Output. This entry refers to the type of output that the user *always* gets as a result of the job.
5. Output options available. This entry refers to the type of output the user can get by using output option (OPTN) cards.
6. Required user assignments. This entry describes any additional logical file assignments that the user must make to perform the job.
7. Control cards. This entry describes the method of punching any required control cards and output option (OPTN) cards.

Notes: 1. Any logical file assumed assignment can be changed by using an ASGN card. (See *Changing File Assignments.)*

2. NOTE and PAUSE cards can be placed between, but not within job decks.

Conventional Assembly

A conventional assembly refers to the results normally associated with assembling an object program. All information concerned with required control cards and the manner of punching the control cards is included in the following section.

Purpose of Job	Processor Job	Input			Output									
					Documentation			Object Program				Messages		
		Source Program	Autocoder Text	Object Program (card deck or coreload format)	CTL Card Diagnostics (if CTL card contain errors)	Source Program Diagnostics and Label Table or Cross-reference Listing**	Program Listing	Condensed-Loader Format	Self-Loading Format	Coreload Format	Autocoder Text	Location of Coreload	Location of Text	Resequenced Source Deck
Conventional Assembly	AUTOCODER RUN THRU OUTPUT	YES			YES	YES	YES †	YES✓†	OPT††	OPT		OPT*		OPT
Load-and-Go	AUTOCODER RUN THRU EXECUTION	YES			YES	YES	YES			YES		YES		
Delayed Execution	EXECUTION RUN			YES										
Partial Processing	AUTOCODER RUN	YES			YES	YES					YES		YES	
	OUTPUT RUN		YES				OPT	OPT	OPT	OPT		OPT*		OPT
	OUTPUT RUN THRU EXECUTION		YES				YES			YES		YES		

** Depend on CTL card specifications
† Additional listings and condensed-loader decks are available.
* Message is associated with the Coreload option.
†† Specified in CTL card or Output OPTN card.
✓ Unless the self-loading format is specified in the CTL card.

Figure 6. Processor Jobs

Autocoder Run Thru Output

This is the type of run that results in a conventional assembly.

Assumed Input Device: INPUT file on READER 1.

Input: Source program.

Assumed Output Devices: LIST file on PRINTER 2, MESSAGE file on PRINTER 2, OUTPUT file on PUNCH 1 (1442) or PUNCH 4 (1402).

Output:

1. CTL diagnostic messages, if errors are sensed.
2. Source-statement diagnostic messages, unless the CTL card specifies that the diagnostic phases be omitted.
3. Cross-reference listing, label table, or neither, depending on CTL card specification.
4. Program listing (100-character or 120-character).
5. Object program in the condensed-loader format (six-card loader for 1401 or 1460, seven-card loader for 1440), or an object program in the self-loading format, if specified in the CTL card.

Output Options Available:

1. Additional program listing. To obtain this option, use a LIST OPTN card.
2. Object program in the condensed-loader format. To obtain this option, use a PUNCH OPTN card.
3. Object program in the self-loading format. To obtain this option, use a PUNCH OPTN card.
4. Object program in the coreload format and a message specifying the START and END addresses of the CORELOAD file. To obtain this option, use a CORELOAD OPTN card.

Required User Assignments: If the object program is to be punched into cards, the user does not have to make any file assignments. However, if an object program in the coreload format is desired, the CORELOAD file must be assigned before the job is performed. Use a CORELOAD ASGN card to define the file.

Control Cards:

1. The RUN card is the only control card required for a conventional assembly. Punch the RUN card in the following manner:

Columns	*Contents*
6-14	AUTOCODER
16-18	RUN
21-24	THRU
26-31	OUTPUT

2. The following cards are punched only if the user wishes any of the available output options. Any one or all of these cards can be used with a conventional assembly.

   a. LIST OPTN for an additional program listing. Punch the LIST OPTN card in the following manner:

Columns	*Contents*
6-9	LIST
16-19	OPTN
21-22	Number of lines (01-99) per page

(If left blank, user's carriage control tape will regulate listing.)

Note: If the SPCE 2 control statement is used with the LIST OPTN card, the maximum number of available statements per page is 98. If SPCE 3 control statement is used, the maximum number is 97.

b. PUNCH OPTN for card formats. Use the PUNCH OPTN card only if an additional object deck is desired. Punch the card in the following manner:

Columns	*Contents*
6-10	PUNCH
16-19	OPTN
21	S if the self-loading format is desired; blank if the condensed-loader format is desired

c. CORELOAD OPTN for coreload format. Punch the CORELOAD OPTN card in the following manner:

Columns	*Contents*
6-13	CORELOAD
16-19	OPTN

If the CORELOAD OPTN card is used, a CORELOAD ASGN card, which precedes the RUN card, must be used to define the CORELOAD file. Punch the CORELOAD ASGN card in the following manner:

Columns	*Contents*
6-13	CORELOAD
16-19	ASGN
21-57	1301 UNIT *n*, START *nnnnnn*, END *nnnnnn* or 1311 UNIT *n*, START *nnnnnn*, END *nnnnnn*

The value of *n* represents the number of the disk unit, and can be 0, 1, 2, 3, or 4; *nnnnnn* represents a disk address. The limits specified must define an area large enough to contain the object program. When punching the CORELOAD ASGN card, blanks must be present in columns 21-57 where indicated in the format.

d. RESEQ OPTN for a resequenced source deck. Punch the RESEQ OPTN card in the following manner:

Columns	*Contents*
6-10	RESEQ
16-19	OPTN

Arrangement. The arrangement of input cards is shown in Figure 7. OPTN cards can be in any order.

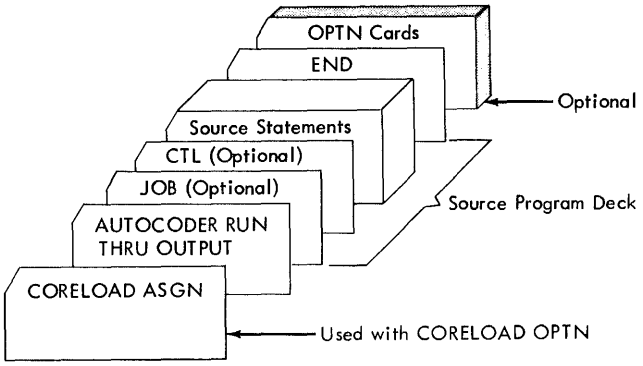

Figure 7. Conventional Assembly

Note: If the PUNCH OPTN and/or the RESEQ OPTN is chosen and the punch is 1442 and the CONTROL and OUTPUT files are assigned to the same unit, the user must follow each of the OPTN cards with a sufficient number of blank cards.

Load-and-Go

This is similar to the standard load-and-go operation. The difference is that the object program is available on the CORELOAD file for delayed execution. All information regarding required control cards and the manner of punching these control cards is contained in the following section.

Autocoder Run Thru Execution

This is the type of run that results in load-and-go.

Assumed Input Device: INPUT file on READER 1.

Input: Source program.

Assumed Output Devices: MESSAGE file on PRINTER 2, LIST file on PRINTER 2.

Output:

1. CTL diagnostic messages if errors are sensed.
2. Source-statement diagnostic messages, unless the CTL card specifies that the diagnostic phases be omitted.
3. Cross-reference listing, label table, or neither, depending on CTL card specification.
4. Program listing.
5. Object program in the coreload format and a message specifying the START and END addresses of the program that is stored on the CORELOAD file in disk storage.

Output Options Available: None.

Required User Assignments: The CORELOAD file must be defined by the user before the job is performed. Use a CORELOAD ASGN card specifying the START and END addresses of the CORELOAD file to define the file.

Additional Results: The object program is loaded into core storage and control is transferred to it.

Control Cards: Two control cards, a RUN and an ASGN card, are required for the load-and-go option.

1. A CORELOAD ASGN card, which precedes the RUN card, must be used to define the CORELOAD file. Punch the CORELOAD ASGN card in the following manner:

Columns	*Contents*
6-13	CORELOAD
16-19	ASGN
21-57	1301 UNIT *n*, START *nnnnnn*, END *nnnnnn* or 1311 UNIT *n*, START *nnnnnn*, END *nnnnnn*

The value *n* is the number of the disk unit, and can be 0, 1, 2, 3, or 4; *nnnnnn* represents a disk address. The limits specified must define an area large enough to contain the object program. When punching the CORELOAD ASGN card, blanks must be present in columns 21-57 where indicated in the format.

2. Punch the required RUN card in the following manner:

Columns	*Contents*
6-14	AUTOCODER
16-18	RUN
21-24	THRU
26-34	EXECUTION

Arrangement: The arrangement of input cards is shown in Figure 8.

Delayed Execution

This job enables the user to execute an object program under the control of the Autocoder System.

Note: If the SYSCL macro was not included in the source program, control will not be returned to the Autocoder System after execution of the object program.

Execution Run

This is the type of run that is used when an object program is executed in a stack of jobs.

Assumed Input Device: INPUT file on READER 1.

Input: Object program.

Assumed Output Devices: Not applicable.

Output: Not applicable.

Output Options Available: Not applicable.

Required User Assignments: If the input for the run is an object-program card deck in either the condensed-loader or self-loading format, no INPUT ASGN card is required. However, if the input for the run is an object program in the coreload format, the INPUT file must be defined before the job is performed. Use an INPUT ASGN card to define the file.

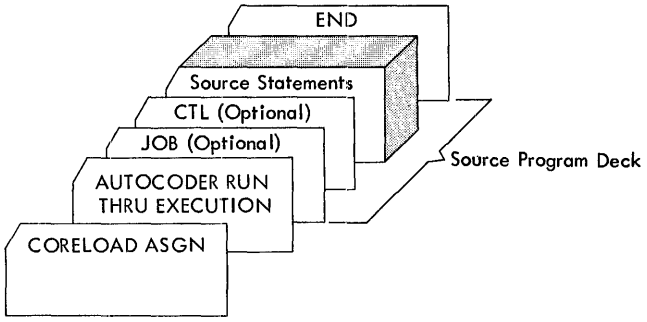

Figure 8. Load-and-Go

Control Cards:

1. An INPUT ASGN card is required if the object program is in the coreload format. The INPUT ASGN card precedes the RUN card. Punch the INPUT ASGN card in the following manner:

Columns	*Contents*
6-10	INPUT
16-19	ASGN
21-57	1301 UNIT *n*, START *nnnnnn*, END *nnnnnn* or 1311 UNIT *n*, START *nnnnnn*, END *nnnnnn*

The value *n* is the number of the disk unit, and can be 0, 1, 2, 3, or 4; *nnnnnn* represents a disk address. The START and END disk addresses of the object program are given in the message printed at the completion of the operation that built the CORELOAD file. When punching the INPUT ASGN card, blanks must be present in columns 21-57 where indicated in the format.

2. Punch the required RUN card in the following manner:

Columns	*Contents*
6-14	EXECUTION
16-18	RUN
21-?	[*JOB card operand*]
76-80	[*JOB card identification*]

If the object program is in the coreload format, the JOB card information in the RUN card (if any is punched) is compared with the JOB card information in the object program (on disk) to ensure that the correct disk address has been specified in the INPUT ASGN card.

Arrangement: The arrangement of input cards is shown in Figures 9 and 10. If an INPUT ASGN card is used, it must precede the RUN card.

Note: For 1402 assign the CONTROL and INPUT files to READER 0 to insure that all input cards will be in the NR pocket.

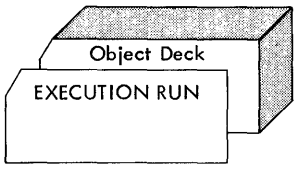

Figure 9. Delayed Execution (Object Deck)

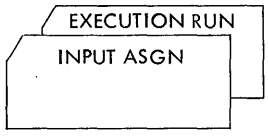

Figure 10. Delayed Execution (Coreload Format)

Partial Processing

The Autocoder System permits the user to select certain processors in the Autocoder System. This feature enables the user to save the Autocoder text for future output processing. All information regarding required control cards and the manner of punching these control cards is contained in the following section.

Autocoder Run

The result of this job is Autocoder text. The text requires processing by the Output processor because it is not in a usable form.

Assumed Input Device: INPUT file on READER 1.

Input: Source program.

Assumed Output Devices: MESSAGE file on PRINTER 2, LIST file on PRINTER 2.

Output:

1. CTL diagnostic messages, if errors are sensed.
2. Source-statement diagnostic messages, unless the CTL card specifies that the diagnostic phases are to be omitted.
3. Cross-reference listing, label table, or neither, depending on CTL card specification.
4. Autocoder text and a message specifying the START address of the text.

Output Options Available: None.

Required User Assignments: Because the result of processing is Autocoder text, an area (OUTPUT file) in disk storage must be defined. The OUTPUT file must be defined before the job is performed. Use an OUTPUT ASGN card to define the file.

Control Cards:

1. An OUTPUT ASGN card, which precedes the RUN card, must be used to define the OUTPUT file because the Autocoder text is written in disk storage. Punch the OUTPUT ASGN card in the following manner:

Columns	*Contents*
6-11	OUTPUT
16-19	ASGN
21-57	1301 UNIT *n*, START *nnnnnn*, END *nnnnnn* or, 1311 UNIT *n*, START *nnnnnn*, END *nnnnnn*

The value *n* indicates the number of the disk unit, and can be 0, 1, 2, 3, or 4; *nnnnnn* represents a disk address. The limits specified must define an area large enough to contain the Autocoder text. When punching the OUTPUT ASGN card, blanks must be present in columns 21-57 where indicated in the format.

2. Punch the required RUN card in the following manner:

Columns	*Contents*
6-14	AUTOCODER
16-18	RUN

Arrangement: The arrangement of input cards is shown in Figure 11.

Output Run

This job enables the user to process the Autocoder text produced by an AUTOCODER RUN and to specify the kind(s) of output he desires.

Assumed Input Device: This must be an area in disk storage defined by the user.

Input: Autocoder text.

Assumed Output Devices: MESSAGE file on PRINTER 2, LIST file on PRINTER 2, OUTPUT file on PUNCH 1 (1442) or PUNCH 4 (1402).

Output: The kind of output must be specified by the user.

Output Options Available:

1. PUNCH option—an object program in the condensed-loader or self-loading format. To obtain this option, use a PUNCH OPTN card.
2. CORELOAD option—an object program in the core-load format and a message specifying the START and END addresses of the program. To obtain this option, use a CORELOAD OPTN card.
3. LIST option—a 100-character or 120-character program listing. To obtain this option, use a LIST OPTN card.
4. RESEQ option—a resequenced source deck. To obtain this option, use a RESEQ OPTN card.

Required User Assignments:

1. The INPUT file must be defined before the job is performed. Use an INPUT ASGN card to define the file.

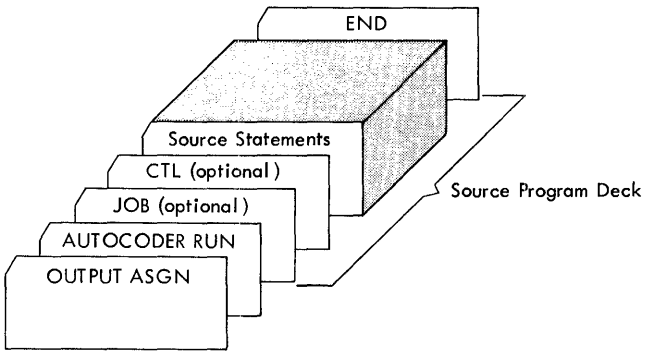

Figure 11. AUTOCODER RUN

2. If an object deck in the condensed-loader format is desired, no OUTPUT file need be defined. (The OUTPUT file is assumed to be PUNCH 1 for 1442 and PUNCH 4 for 1402.) However, if an object program in the coreload format is desired, the CORELOAD file must be defined before the job is performed. Use a CORELOAD ASGN card to define the file.

Control Cards:

1. An INPUT ASGN card is required because the Autocoder text is in disk storage. The INPUT ASGN card precedes the RUN card. Punch the INPUT ASGN card in the following manner:

Columns	*Contents*
6-10	INPUT
16-19	ASGN
21-57	1301 UNIT *n*, START *nnnnnn*, END *nnnnnn* or 1311 UNIT *n*, START *nnnnnn*, END *nnnnnn*

The value *n* indicates the number of the disk unit, and can be 0, 1, 2, 3, or 4; *nnnnnn* represents a disk address. The START address of the Autocoder text is given in the message printed at the beginning of an AUTOCODER RUN. The END address is given in the message printed when the disk OUTPUT file is closed. When punching the INPUT ASGN card, blanks must be present in columns 21-57 where indicated in the format.

2. Punch the required RUN card in the following manner:

Columns	*Contents*
6-11	OUTPUT
16-18	RUN

3. Output option cards:
   a. Punch the PUNCH OPTN card in the following manner:

Columns	*Contents*
6-10	PUNCH
16-19	OPTN
21	S if the self-loading format is desired. Blank if the condensed-loader format is desired.

   b. Punch CORELOAD OPTN card in the following manner:

Columns	*Contents*
6-13	CORELOAD
16-19	OPTN

If the CORELOAD OPTN card is used, a CORELOAD ASGN card, which precedes the RUN card, must be used to define the CORELOAD file. Punch the CORELOAD ASGN card in the following manner:

Columns	*Contents*
6-13	CORELOAD
16-19	ASGN
21-57	1301 UNIT *n*, START *nnnnnn*, END *nnnnnn* or 1311 UNIT *n*, START *nnnnnn*, END *nnnnnn*

The value *n* indicates the number of the disk unit, and can be 0, 1, 2, 3, or 4; *nnnnnn* represents a disk address. The limits specified must define an area large enough to contain the object program. When punching the CORELOAD ASGN card, blanks must be present in columns 21-57 where indicated in the format.

   c. Punch the LIST OPTN card in the following manner:

Columns	*Contents*
6-9	LIST
16-19	OPTN
21-22	Number of statements (01-99) per page. (If left blank, user's carriage control tape will regulate listing.)

Note: If the SPCE 2 control statement is used with the LIST OPTN card, the maximum number of available statements per page is 98. If SPCE 3 control statement is used, the maximum number is 97.

   d. Punch the RESEQ OPTN card in the following manner:

Columns	*Contents*
6-10	RESEQ
16-19	OPTN

Arrangement: The arrangement of input cards is shown in Figure 12. At least one option card is required indicating the type of output. The option cards can be in any order.

Note: If the PUNCH OPTN and/or the RESEQ OPTN is chosen, and the punch is 1442 and the CONTROL and OUTPUT files are assigned to the same unit, the user must follow each card with a sufficient number of blank cards.

Output Run Thru Execution

This job enables the user to process the Autocoder text and execute the resulting object program.

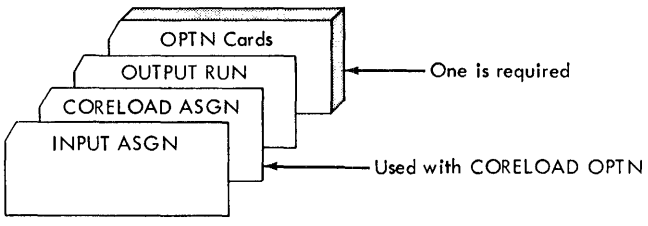

Figure 12. OUTPUT RUN

Assumed Input Device: The INPUT file must be an area in disk storage that is defined by the user indicating the location of the Autocoder text.

Input: Autocoder text.

Assumed Output Devices: MESSAGE file on PRINTER 2 and LIST file on PRINTER 2.

Output:

1. Program listing (100-character or 120-character).
2. Object program in the coreload format and a message specifying the START and END addresses of the program.

Output Options Available: None.

Required User Assignments: The INPUT and CORELOAD files must be defined before the job is performed. Use an INPUT ASGN card and a CORELOAD ASGN card to define the files.

Additional Results: The object program is loaded into core storage and control is transferred to it.

Control Cards:

1. An INPUT ASGN card is required because the Autocoder text is in disk storage. The INPUT ASGN card precedes the RUN card. Punch the INPUT ASGN card in the following manner:

Columns	*Contents*
6-10	INPUT
16-19	ASGN
21-57	1301 UNIT *n*, START *nnnnnn*, END *nnnnnn* or 1311 UNIT *n*, START *nnnnnn*, END *nnnnnn*

   The value *n* indicates the number of the disk unit, and can be 0, 1, 2, 3, or 4; *nnnnnn* represents a disk address. The disk address of the Autocoder text is given in the message printed at the completion of an AUTOCODER RUN. When punching the INPUT ASGN card, blanks must be present in columns 21-57 where indicated in the format.

2. A CORELOAD ASGN card is required because the object program is written in disk storage on the CORELOAD file. The CORELOAD ASGN card precedes the RUN card. Punch the CORELOAD ASGN card in the following manner:

Columns	*Contents*
6-13	CORELOAD
16-19	ASGN
21-57	1301 UNIT *n*, START *nnnnnn* END *nnnnnn* or 1311 UNIT *n*, START *nnnnnn*, END *nnnnnn*

   The value *n* indicates the number of the disk unit, and can be 0, 1, 2, 3, or 4; *nnnnnn* represents a disk address. The limits specified must define an area large enough to contain the object program. When punching the CORELOAD ASGN card, blanks must be present in columns 21-57 where indicated in the format.

3. Punch the required RUN card in the following manner:

Columns	*Contents*
6-10	OUTPUT
16-18	RUN
21-24	THRU
26-34	EXECUTION

Arrangement: The arrangement of the input cards is shown in Figure 13. The ASGN cards can be in any order.

Changing File Assignments

Each logical file defined by the Autocoder System, with the exception of the SYSTEM and CORELOAD files, is assigned to a specific input/output device by the System Control Program. These assignments can be changed by using ASGN cards (and, in certain instances, INIT cards – see *INIT Card*). The uses of the logical files should be considered when deciding file assignments.

Figures 14 and 15 illustrate the uses of all the logical files, except the SYSTEM file, that are required for processor jobs. The SYSTEM file, on which the Autocoder System resides, is required for all System operations.

In addition to the cards listed in Figure 14, the CONTROL file contains:

1. The 1402 or 1442 Card Boot, which is the first set of cards required for stack processing. (A stack consists of one or more jobs.)
2. All ASGN cards. Some ASGN cards are required for particular processor jobs. Other ASGN cards cause file-assignment changes which make it possible to utilize the input/output devices not included in the assumed file assignments.
3. INIT, NOTE and PAUSE cards, which may be inserted between jobs.
4. A HALT card, which must be the last card in a stack.

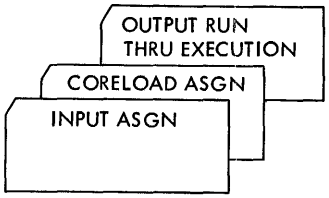

Figure 13. OUTPUT RUN THRU EXECUTION

Processor Job		Operation Files	Internal Files		
	CONTROL	MESSAGE	WORK1	WORK2	WORK3
AUTOCODER RUN THRU OUTPUT	CORELOAD ASGN card if the CORELOAD OPTN card is used. RUN card.	CTL diagnostics, if the CTL card contains errors. Source statement diagnostics, if specified (CTL card). CORELOAD OUTPUT COMPLETE ON {1311 / 1301} UNIT n, START nnnnnn, END nnnnnn CORELOAD HEADER — (52 positions), ID — (5 positions) These messages are printed if the CORELOAD ASGN and CORELOAD OPTN cards are in the CONTROL file.	Intermediate and final results of Autocoder processing (Autocoder text).	References for cross reference listing, if specified (CTL card).	Used by the Macro Generator and the Autocoder Processor. Labels for cross reference listing or label table. Intermediate results of Output processing if the CORELOAD ASGN and CORELOAD OPTN cards are in the CONTROL file.
AUTOCODER RUN THRU EXECUTION	CORELOAD ASGN card. RUN card.	CTL diagnostics, if the CTL card contains errors. Source statement diagnostics, if specified (CTL card). CORELOAD OUTPUT COMPLETE ON {1311 / 1301} UNIT n, START nnnnnn, END nnnnnn CORELOAD HEADER — (52 positions), ID — (5 positions)	Intermediate and final results of Autocoder processing (Autocoder text).	References for cross reference listing, if specified (CTL card).	Used by the Macro Generator and the Autocoder Processor. Labels for cross reference listing or label table. Intermediate results of Output processing.
EXECUTION RUN (punched-card object program)	RUN card.				
EXECUTION RUN (object program in coreload format)	INPUT ASGN card. RUN card.	INPUT FILE STARTS ON {1311 / 1301} UNIT n AT ADDRESS nnnnnn			
AUTOCODER RUN	OUTPUT ASGN card. RUN card.	CTL diagnostics, if the CTL card contains errors. Source statement diagnostics, if specified (CTL card). OUTPUT FILE STARTS ON {1311 / 1301} UNIT n AT ADDRESS nnnnnn		References for cross reference listing, if specified (CTL card).	Used by the Macro Generator and the Autocoder Processor. Labels for cross reference listing or label table.
OUTPUT RUN	INPUT ASGN card. CORELOAD ASGN card if the CORELOAD OPTN card is used. RUN card. Output OPTN card(s). (At least one must be used.)	INPUT FILE STARTS ON {1311 / 1301} UNIT n AT ADDRESS nnnnnn CORELOAD OUTPUT COMPLETE ON {1311 / 1301} UNIT n, START nnnnnn, END nnnnnn CORELOAD HEADER — (52 positions), ID — (5 positions) The last two messages are printed if the CORELOAD ASGN and CORELOAD OPTN cards are in the CONTROL file.			Intermediate results of Output processing if the CORELOAD ASGN and CORELOAD OPTN cards are in the CONTROL file.
OUTPUT RUN THRU EXECUTION	INPUT ASGN card. CORELOAD ASGN card. RUN card.	INPUT FILE STARTS ON {1311 / 1301} UNIT n AT ADDRESS nnnnnn CORELOAD OUTPUT COMPLETE ON {1311 / 1301} UNIT n, START nnnnnn, END nnnnnn CORELOAD HEADER — (52 positions), ID — (5 positions)			Intermediate results of Output processing.

Figure 14. Use of Operation and Internal Logical Files

The contents of the MESSAGE file, as shown in Figure 14, provide job documentation. Other messages that also appear on the MESSAGE file are: diagnostics relating to operating procedures, instructions to the machine operator, ASGN card images.

As shown in Figure 14, the WORK files contain intermediate results of processing. WORK1 is required for Autocoder processing. If AUTOCODER RUN THRU OUTPUT or AUTOCODER RUN THRU EXECUTION is specified, WORK1 becomes the OUTPUT file from the Autocoder processor and the INPUT file to the Output processor.

WORK2 is used by the Autocoder processor if a cross-reference listing is requested by the CTL card. After the listing has been built on WORK2, it is transferred to the LIST file.

WORK3 is used by the Macro Generator and the Autocoder processor. It is also required whenever the CORELOAD file is to be used. The CORELOAD file is used if THRU EXECUTION is specified in a RUN card, or if a CORELOAD OPTN card is used for an AUTOCODER RUN THRU OUTPUT or OUTPUT RUN. If THRU EXECUTION is specified in a RUN card, the CORELOAD file becomes the INPUT file to the Execution processor.

Thus, the WORK1 and CORELOAD files act as transition files between processors when a THRU option is specified.

Processor Job	External Files				
	INPUT	OUTPUT	LIST	CORELOAD	LIBRARY
AUTOCODER RUN THRU OUTPUT	Source program.	Object program in condensed-loader format, unless self-loading format is specified in the CTL card. Object program(s) in the self-loading format, if specified (PUNCH OPTN or CTL card). Additional object program(s) in condensed-loader format, if specified (PUNCH OPTN). Resequenced source deck(s) if specified (RESEQ OPTN):	Label table or cross-reference listing if specified (CTL card). Program listing. Additional program listing(s), if specified (LIST OPTN).	Object program in the coreload format, if specified (CORE-LOAD OPTN).	Used during macro generation.
AUTOCODER RUN THRU EXECUTION	Source program.		Label table or cross-reference listing if specified (CTL card). Program listing.	Object program in coreload format.	Used during macro generation.
EXECUTION RUN	Object program				
AUTOCODER RUN	Source program	Intermediate and final results of Autocoder processing (Autocoder text).	Label table or cross-reference listing if specified (CTL card).		Used during macro generation.
OUTPUT RUN	Autocoder text.	Object program(s) in condensed-loader and/or self-loading format, if specified (PUNCH OPTN). Resequenced source deck(s), if specified (RESEQ OPTN).	Program listing(s), if specified (LIST OPTN).	Object program in coreload format, if specified (CORE-LOAD OPTN).	
OUTPUT RUN THRU EXECUTION	Autocoder text.		Program listing.	Object program in coreload format.	

Figure 15. Use of External Logical Files

Preparing ASGN Cards

ASGN cards enable the user to change file assignments for one or more jobs in a stack. The general format for an ASGN card is:

file-name ASGN {*device* / OMIT}

The *file-name* is the specific logical file; *device* is the input/output unit to which the logical file is assigned.

The assumed file assignments and ASGN card formats relating to specific files are shown in Figure 16. Valid device entries are shown in Figure 17.

Leave a blank between items in the operand field as shown in Figure 16. If, for example, the OUTPUT file is to be assigned to disk area 004000 through 004799 on 1301 unit 1, the user would code the ASGN card for punching as shown in Figure 18. The END address to be punched is the address of the next available sector, not the address of the last sector to be used.

File Considerations

CONTROL File and INPUT File. If both the CONTROL and INPUT files are assigned to the reader, the assignments must be identical. For example, if the system is a 1440 and the CONTROL file is assigned to READER 1, the INPUT file must also be assigned to READER 1.

MESSAGE File and LIST File. If both the MESSAGE and LIST files are assigned to the printer, the assignments must be identical. For example, if the system is a 1401 and the MESSAGE file is assigned to PRINTER 2, the LIST file must also be assigned to PRINTER 2.

WORK1 File. The disk area required for the WORK1 file depends upon the number of statements in the source program after macro generation and upon the number of literals used in the program. The user must allow one sector for each statement plus one sector for each unique literal.

WORK2 *File.* This file must contain at least 200 sectors for every 600 references to labels and area-defining literals in the source program. Storage from this file is used in blocks of 200 sectors. If the amount of storage left in the file is less than 200 sectors and data remains to be stored in the file, the assembler will halt, indicating that an area is too small.

WORK3 *File.* This file requires a minimum of 300 sectors of disk storage. Additional sectors could be required, depending on the number of labels and area-defining literals used in the source program. At least 200 sectors are required for every 600 different labels and area-defining literals used in the source program. However, in storing labels and area-defining literals, the assembler uses WORK3 storage in sections of 200 sectors. Thus, if the assembler fills 200 sectors, and more labels or area-defining literals remain to be stored, at least 200 sectors must remain in the WORK3 area or the

ASGN Card Format			Assumed Assignment	Remarks
Label Field (Columns 6-15)	Operation Field (Columns 16-20)	Operand Field (Columns 21-72)		
SYSTEM	ASGN	{1311 UNIT n / 1301 UNIT 0}	1311 unit -- user-assigned 1301 unit -- must be assigned to UNIT 0	The SYSTEM ASGN card is the only required ASGN card. It must follow the Card Boot in a stack of jobs. Any other SYSTEM ASGN cards in the stack are invalid. If the user desires that the Autocoder System use less than the number of core-storage positions available in the processor machine, punch a comma in column 32 and 4K, 8K, 12K, or 16K beginning in column 34.
CONTROL	ASGN	{READER n / CONSOLE PRINTER}	READER 1	
MESSAGE	ASGN	{PRINTER n / CONSOLE PRINTER}	PRINTER 2	When the MESSAGE file is assigned to the CONSOLE PRINTER, carriage control characters used with the 1403 or 1443 printer may appear in the message.
LIST	ASGN	{PRINTER n / 1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn / OMIT}	PRINTER 2	If the LIST file is assigned to PRINTER 1 (1403), the Output processor develops a 100-character program listing.
INPUT	ASGN	{READER n / 1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn}	READER 1	
OUTPUT	ASGN	{PUNCH n / 1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn / OMIT}	PUNCH 4 (1401 and 1460) PUNCH 1 (1440)	
LIBRARY	ASGN	{1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn}	{1301 / 1311} UNIT 0, START 012900, END 019980	1311 is assumed if the SYSTEM file is assigned to 1311; 1301 is assumed if the SYSTEM file is assigned to 1301.
WORK1	ASGN	{1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn}	{1311 / 1301} UNIT 0, START 004800, END 011200	
WORK2	ASGN	{1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn}	{1311 / 1301} UNIT 0, START 011200, END 012400	
WORK3	ASGN	{1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn}	{1311 / 1301} UNIT 0, START 012400, END 012900	
CORELOAD	ASGN	{1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn / OMIT}	OMIT	

NOTE: If the user's system contains Autocoder and COBOL, the WORK1 assumed assignment is changed from START 004800, END 011200 to START 007200, END 010400. The assumed assignments for WORK2 and WORK3 remain the same.

Figure 16. ASGN Card Formats and Assumed Assignments

assembler will halt indicating that an area is too small.

Note: If the SYSTEM file is on a 1311 drive other than drive 0 and drive 0 is not on-line, the user must change the WORK1, WORK2, and WORK3 file assignments because the Autocoder System assumes that the WORK files are on 1311 UNIT 0.

CORELOAD File. The disk area required for an object program in the coreload format depends upon the type of statements used in the source program, the number of characters (instructions and data) in the object program, and the number of loading instructions developed by the Output processor.

When the Output processor transfers the object program to the CORELOAD file, it builds as many full 90-character sectors as possible. These 90-character sectors contain data (such as a constant defined by a DCW statement) and assembled instructions (such as M 411 199). Each DA, DS, ORG, LTORG, XFR, and EX statement causes the processor to begin building a new set of 90-character sectors.

Use the following as a guide for approximating the disk area required:

1. One sector for the HEADER record. This record contains the operand and the identification from the last JOB card.
2. One sector for every 24 word marks specified in each set of subsequent DA entries.
3. One sector for each record mark specified in each DA header.
4. Two sectors for a DA header that specifies that the defined area(s) be cleared.
5. One sector for each ,G specified in the source program (DA header, DCW, and DC statements).
6. One sector for each EX, XFR, and END statement. Each of these sectors contains a branch to the program at object time.
7. One sector for every 90 object-program charac-

Device Entry and Values of n and nnnnnn	Remarks
{1311 / 1301} UNIT n, START nnnnnn, END nnnnnn n is the number of the disk unit, and can be 0, 1, 2, 3, or 4; nnnnnn is a disk address.	The END address is the address of the next available sector. The values of nnnnnn must adhere to the following rules: 1. WORK1 file. If the disk unit is 1311, the START address must be a multiple of 200. If the disk unit is 1301, the START address must be a multiple of 800. The END address (1311 and 1301) must be a multiple of 40. 2. WORK2 and WORK3 files. The START address (1311 and 1301) of each file must be a multiple of 100. (A START address that is a multiple of 200 results in the fastest assembly.) The END address (1311 and 1301) of each file must be a multiple of 10. 3. LIBRARY file. The START and END addresses (1311 and 1301) must be multiples of 20. 4. OUTPUT file. It is not necessary to specify that this file start or end at any particular multiple. However, Autocoder will only use the file if it begins at a multiple of 40. In each of the first three cases, if the rules are violated, the system automatically narrows in the disk area to an area that does adhere to the rules. Incorrect addresses are not automatically corrected for the OUTPUT file.
READER n For 1402, n can be 0, 1, or 2. For 1442, n can be 1 or 2.	 For 1402, n represents the pocket into which the cards are stacked. For 1442 and 1444, n represents the number of the unit.
PUNCH n For 1402, n can be 0, 4, or 8. For 1442, n can be 1 or 2 For 1444, n must be 3.	
PRINTER n n can be 1 or 2	n represents the number of print positions available on the 1403 or 1443. For 1403, a 1 indicates 100 positions and a 2 indicates 132 positions. For 1443, a 1 indicates 120 positions and a 2 indicates 144* positions. * Only 132 print positions are used by the Autocoder System.
CONSOLE PRINTER	The console printer for the control file must be an IBM 1447 without a buffer feature or an IBM 1407. An IBM 1447 with a buffer feature can be used for the message file, although the buffer feature is not used.
OMIT	Select this option when the file is not to be used by the Autocoder System. LIST, OUTPUT, and CORELOAD are the only files that can be omitted.

Figure 17. Valid Device Entries

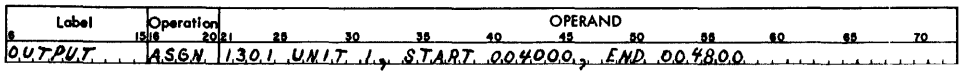

Figure 18. Coding for an OUTPUT ASGN Card

ters that precede each DA, DS, ORG, LTORG, XFR, EX, and END statement. Approximate the number of 90-character sectors in each set by:

a. Allowing seven object-program characters for each imperative and declarative, except DS and DA, statements.
b. Approximating the number of generated statements associated with each macro instruction and multiplying the approximation by 7.

8. One sector for the first ten object-program characters in each set of 90-character sectors. Because the first ten positions of each set of 90-character sectors must contain a disk control word, the processor builds a sector that contains the first ten object-program characters in a set and the instructions that load the ten characters at object time.
9. At least one sector for the last group of object program characters before each DS, DA, ORG,

LTORG, XFR, EX, and END statement if the group contains fewer than 90 characters. A maximum of 50 object-program characters can be contained in one sector because the first portion of the sector contains instructions that load the object-program characters at object time.

10. One sector to load each set of 90-character sectors at object time.

To ensure that a sufficient disk area is allotted for the CORELOAD file, the programmer should use his source-program coding sheet to approximate the number of sectors required. He should add five sectors to his approximation to allow for cylinder overflow.

OUTPUT File. This file must be assigned to a disk area for an AUTOCODER RUN because the Autocoder text (100-character records) must be on disk for Output processing.

The disk area required for the Autocoder text is determined in the same manner as the area required for WORK1.

Note: Do not assign the OUTPUT file to a disk area for an AUTOCODER RUN THRU OUTPUT or for an OUTPUT RUN.

LIBRARY File. The method for determining the disk area required for a LIBRARY file is given in *Preparing Library Jobs.*

A LIBRARY file is required for an AUTOCODER RUN, an AUTOCODER RUN THRU OUTPUT, and an AUTOCODER RUN THRU EXECUTION. The user must be sure to include a LIBRARY ASGN card in the stack if Autocoder processing is to be performed and the LIBRARY file assignment (unit number and/or limits) differs from that assumed by the Autocoder System.

Building a LIBRARY file and transferring routines to it are described under *Preparing Library Jobs.*

Note: If it is necessary to rebuild the Preprocessor, the user can avoid destroying the LIBRARY file that is within the limits assumed by the Autocoder System. Place a *dummy* LIBRARY ASSGN card ahead of the AUTOCODER RUN card which is the first card in the section of the System deck labeled AUTOCODER PREPROCESSOR. This *dummy* ASGN card should specify a disk area whose contents need not be saved. For example, the area that is allotted to WORK1 could be specified.

Using ASGN Cards

At the beginning of stack processing, the System Control Program reads a list of assumed assignments into core storage from the SYSTEM file. Each assumed assignment remains in effect until an ASGN card for that file is sensed. Any changed file assignment remains in effect until the next ASGN card for that file, or a HALT card, is sensed.

If a file-assignment change is applicable for an entire stack, place the ASGN card immediately ahead of the first RUN card.

If a file-assignment change is only applicable to a specific job, place the ASGN card immediately ahead of the RUN card for that job. To change the file assignment back to the assumed assignment or to a different assignment, place the ASGN card immediately ahead of the RUN card for the next job that requires the effective file assignment to be changed.

Example. Figure 19 shows the use of ASGN cards. Assume that:

1. The stack consists of Job 1, Job 2, and Job 3.
2. The stack is to be on an IBM 1460 system with IBM 1311 Disk Storage Drives and an IBM 1301 Disk Storage Unit.
3. The System, WORK1, WORK2, WORK3, and LIBRARY files are located on the 1311 unit 1.
4. The 1311 unit 1 and the 1301 unit 0 are on line.
5. ASGN card A specifies SYSTEM ASGN 1311 UNIT 1. A SYSTEM ASGN card is required for each stack of jobs.
6. ASGN cards B, C, D, and E specify, respectively;
   WORK1 ASGN 1311 UNIT 1, START 004800, END 011200.
   WORK2 ASGN 1311 UNIT 1, START 011200, END 012400.
   WORK3 ASGN 1311 UNIT 1, START 012400, END 012900.
   LIBRARY ASGN 1311 UNIT 1, START 012900, END 019980.
   These ASGN cards are required because drive 0 is not on-line. The limits of the files are those assumed by the Autocoder System. Job 1 is an AUTOCODER RUN THRU OUTPUT.
7. ASGN card F specifies:
   OUTPUT ASGN 1301 UNIT 0, START 120000, END 125000.
   This ASGN card changes the assumed OUTPUT file assignment (Punch 4) for Job 2, which is an AUTOCODER RUN.
8. ASGN card G specifies:
   OUTPUT ASGN PUNCH 4.
   This ASGN card changes the user's OUTPUT file assignment back to the assumed OUTPUT file assignment for Job 3, which is an AUTOCODER RUN THRU OUTPUT. Note that an INIT card could not be used to restore the OUTPUT assumed assignment, because an INIT card would restore *all* assumed file assignments.

Batched Files

Batched files are defined as the external files INPUT, OUTPUT, LIST, and CORELOAD whose contents represent

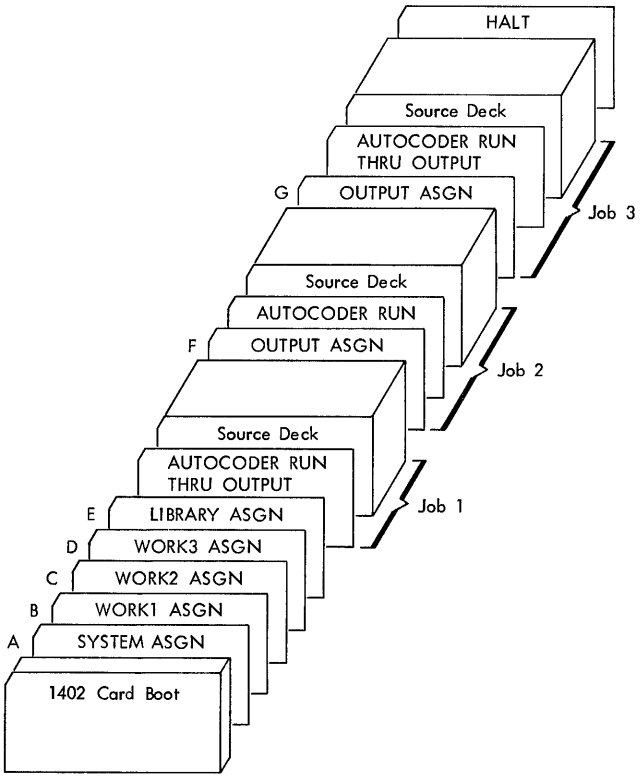

Figure 19. Changing File Assignments

one or more sequential sets of input to or output from the processor(s).

The following example illustrates the advantage of a batched CORELOAD file by describing the building and use of an object-program library.

Procedure: Perform a series of conventional assemblies (AUTOCODER RUN THRU OUTPUT) using the CORELOAD option. Follow the option card for each job with a NOTE card that contains the program identification. Record the messages that are printed during processing.

Result: The programs are assembled and batched (stored sequentially) in the CORELOAD file. After each program has been transferred to disk storage, a message specifying the START and END addresses of the program in disk storage is printed. The information punched in the NOTE card is printed immediately after the message. The inclusion of the NOTE card ensures accurate documentation.

Future Use: When the object programs are required:

1. Refer to the messages that state the disk location of the programs.
2. Prepare INPUT ASGN card(s) using the information supplied in the message. The file containing the programs becomes the INPUT file for one or more EXECUTION RUN jobs.
   a. If the programs are to be selected and executed sequentially, only one INPUT ASGN card is required for the stack because the programs are batched.
   b. If the programs are to be selected randomly and executed, an INPUT ASGN card is required for each job.
3. Prepare an EXECUTION RUN card for each job.
4. Perform the stack of jobs as described under *Running a Stack.*

Preparing Library Jobs

Library jobs are associated with the maintenance of an Autocoder LIBRARY file, which is a disk storage file that supports the Autocoder macro facility. The file contains a library table and library routines, such as Autocoder macros and IOCS.

The three library jobs are:

1. *Library build* which enables the user to define a LIBRARY file. A library-build job, performed when the System is built, defines a LIBRARY file on the same disk unit as the SYSTEM file. The limits of this LIBRARY file are 012900 and 019980. Thus, the assumed assignment for the LIBRARY file is 1301 or 1311 UNIT 0, START 012900, END019980.

   After the library-build job has been performed, the LIBRARY file contains the library table and a record that specifies the end-of-library name (99999).

The library table contains the end-of-library name and its disk address.

2. *Library listing* which enables the user to obtain a list of library routines, a list of routine-names, or a punched-card deck that contains all the statements currently in the LIBRARY file.
3. Library change which enables the user to insert routines in a new LIBRARY file or to modify the contents of an existing LIBRARY file. A library-change job, performed when the System is built, transfers the Autocoder macros to the LIBRARY file after the file has been defined by the library-build job.

Capacity of a LIBRARY File

The model statements that make up a library routine are stored in the LIBRARY file in the following manner: the model statement is compressed and high-order blanks are eliminated. The model statements are stored as variable-length records in two-sector blocks. The library table requires twelve sectors of the LIBRARY file.

The first two positions in every statement in the LIBRARY file are used for a record count. The length of a statement, including the record count field, is:

1. BOOL or MATH statement: 18 characters plus operands and comments.
2. Labeled model statement: 18 characters plus operands and comments.
3. Unlabeled model statement with operands and/or comments: 8 characters plus operands and comments.
4. Unlabeled model statement with no operands or comments:
   a. 2 characters plus the operation code if column 6 is blank.
   b. 8 characters if column 6 is not blank.

   Column 6 can contain a special one-character label that is associated with a BOOL statement.

Note. To fully optimize the library area, the user should not leave more than two blanks between the operands and a comment in a model statement because the librarian phase of Autocoder cannot eliminate unnecessary blanks within the statement.

Library Build

Each library-build job defines a LIBRARY file. The LIBRARY file contains a library table and a record that specifies the end-of-library name (99999). The library table contains the end-of-library name and its disk address.

Perform a library change to insert routines in the new LIBRARY file.

The library build enables the user to increase his library facilities by:

1. Defining one or more LIBRARY files within the limits assumed by the Autocoder System. The use of small LIBRARY files reduces the time required for librarian jobs.
2. Reducing the size of the WORK files and extending the LIBRARY file(s) into that area.
3. Defining one or more libraries that are not located on the same disk unit as the SYSTEM file.

The library build can also be used to define a LIBRARY file with the same limits as an existing LIBRARY file. If the user wishes to delete most of the routines in the file, he may find that it is easier to create a new library table and perform a library change to insert his routines in the LIBRARY file, than it is to perform a library change to delete the routines.

If a library build affects a previously defined LIBRARY file, any routines in the LIBRARY file before the build will not be available at the end of the job because the library build destroys the old library table.

The control cards required for the library build job are:

1. A LIBRARY ASGN card which is required if the assignment of the LIBRARY file differs from that assumed by the Autocoder System. This ASGN card is punched in the following manner:

Columns	*Contents*
6-12	LIBRARY
16-19	ASGN
21-57	1301 UNIT *n*, START *nnnnnn*, END *nnnnnn* or 1311 UNIT *n*, START *nnnnnn*, END *nnnnnn*

The value *n* indicates the number of the disk unit and can be 0, 1, 2, 3, or 4; *nnnnnn* represents a disk address. The limits of the new library must also be specified.

2. A RUN card punched in the following manner:

Columns	*Contents*
6-14	AUTOCODER
16-18	RUN

3. An OPTN card punched in the following manner:

Columns	*Contents*
6-15	INITIALIZE
16-19	OPTN

The INITIALIZE OPTN card may be followed immediately by the cards used to place the macros on the library, beginning with the AUTOCODER RUN card, as described in *Library Change*. Figure 20 shows the arrangement of the control cards.

Library Listing

Four kinds of output are available from the library-listing job:

1. A listing of the names of all the routines (macros) in the Autocoder library.
2. A listing of all the entries in every library routine.
3. A listing of the entries in specific library routines.

Sequence numbers of statements in a library routine are listed under the column header ALTER. These sequence numbers should correspond to sequence numbers used in the DELET and INSER statements that are required for library-change operations.

4. A punched-card deck that contains INSER and model statements (one statement per card). Each routine is preceded by an INSER card. All the routines in the LIBRARY file are punched if this option is selected.

Figure 21 shows a listing of the IBM-supplied LDRCL macro. The characters listed under column L are the labels for BOOL instructions.

The control cards for a library listing are:

1. A LIBRARY ASGN card which is required if the assignment of the LIBRARY file differs from that assumed by the Autocoder System. See *Library Build* for the format of the LIBRARY ASGN card.
2. A RUN card punched in the following manner:

Columns	*Contents*
6-14	AUTOCODER
16-18	RUN

3. An OPTN card punched in the following manner:

Columns	*Contents*
6-12	LISTING
16-19	OPTN
21-?	ALL, if all routines are to be listed, or HEADER if all routine-names are to be listed, or PUNCH if all routines are to be punched into cards using a 1402 or 1442 Card-Read Punch, or PUNCH1444 if all routines are to be punched into cards using a 1444 Card Punch, or *Blank* if only specific routines are to be listed.

Note: When the output from a library listing job is a punched-card deck, a hard halt (halt 006) occurs at the completion of the job. Therefore, when used in a stack, consisting of more than one job, a library punching job should be the last job in the stack.

4. Routine-name cards punched in the following manner are required if specific routines are to be listed:

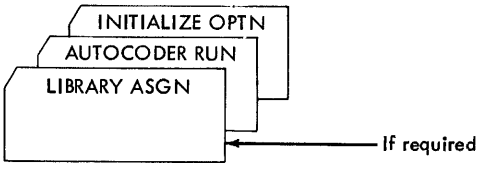

Figure 20. Library Build

```
HEADER   LDRCL

ALTER L LABEL      OP      OPERANDS

0001      ***  RETURN CONTROL TO DISK LOADER    ***
0002               BOOL    A,-#04*-#05
0003               B       3701
0004               NOP
0005               MEND
0006   A           BOOL    A,-#04*#05
0007               B       7701
0008               NOP
0009               MEND
0010   A           BOOL    A,#04*-#05
0011               B       11701
0012               NOP
0013               MEND
0014   A           B       15701
0015               NOP
0016               MEND
```

Figure 21. Library Listing

Columns	*Contents*
16-20	*Name of routine*

The routine-name cards can be in any order.

5. An END card, punched in the following manner, is always required:

Columns	*Contents*
16-18	END

Figure 22 shows the arrangement of the control cards for a library listing of specific routines.

Library Change

Library routines, supplied by IBM or developed by the user, can be added, modified, or deleted. Entries are inserted and/or deleted in collating sequence.

IBM provides a change deck whenever IBM-supplied library routines (macros) should be modified. The change deck includes an AUTOCODER RUN card, a LIBRARY OPTN card, INSER and/or DELET cards, an END card, and cards containing the changes to be made.

The user's change cards can be punched in the disk Autocoder or tape Autocoder format. Library entries in the tape Autocoder format must not contain any □00 notations. The tape Autocoder entries are automatically converted to disk Autocoder format; all condition codes become BOOL statements. The library-change operation cannot process input that contains

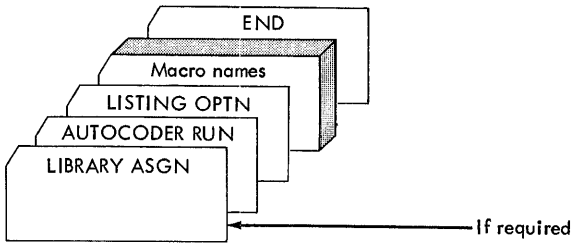

Figure 22. Library Listing

a combination of tape Autocoder and disk Autocoder entries.

For a description of the library statement formats refer to *Autocoder (on Disk) Language Specifications for IBM 1401, 1440, and 1460,* Form C24-3258, and *Autocoder (on Tape) Language Specifications and Operating Procedures for IBM 1401 and 1460,* Form C24-3319.

In addition to the cards containing the changes, the following control cards are required for a library change.

1. A LIBRARY ASGN card which is required if the assignment of the LIBRARY file differs from that assumed by the Autocoder System. See *Library Build* for the format of the LIBRARY ASGN card.
2. A RUN card punched in the following manner:

Columns	*Contents*
6-14	AUTOCODER
16-18	RUN

3. An OPTN card punched in the following manner:

Columns	*Contents*
6-?	LIBRARY or COMPAT
16-19	OPTN
21	1 (if the direct seek feature is available)

If the cards containing the changes are punched in the disk Autocoder format, the LIBRARY OPTN card is required. If the cards are punched in the tape Autocoder format, the COMPAT OPTN card is required.

4. An INSER card is required for each set of insertions. The INSER card is punched in the following manner:

Columns	*Contents*
6-?	Name of the library routine (macro name) to be inserted or modified
16-20	INSER
21-?	Sequence number(s) a. Insertion. Punch *n;* *n* is the number of the statement after which the insertions are to be made. b. Substitution. Punch *n,m;* *n* and *m* are the numbers of the first and last statements to be replaced by the insertions. All statements between and including statement *n* and statement *m* will be deleted, and the insertion will be substituted. Insertions and deletions need not be in a one-to-one correspondence. c. Leave blank if an entire routine is to be inserted or modified. If an entire routine is to be modified, the routine presently in the library will be automatically deleted before the new routine is inserted.

5. A DELET card is required for deletion of a routine or part of a routine. The DELET card is punched in the following manner:

Columns	*Contents*
6-?	Name of library routine (macro name) to be deleted or modified
16-20	DELET
21-?	a. Leave blank if an entire routine is to be deleted. b. If one statement is to be deleted, punch the number of the statement. c. If more than one statement is to be deleted, punch *n,m;* *n* and *m* are the numbers of the first and last statements to be deleted. All statements between and including statement *n* and statement *m* will be deleted.

6. An END card indicates the end of a library-change operation. It is punched in the following manner:

Columns	*Contents*
16-18	END

Figure 23 shows the arrangement of the input cards.

Performing Jobs

Under control of the System Control Program, it is possible to process one or more jobs without operator intervention. For this stack processing to be accomplished, each separate job must be called for by the necessary control cards. A list of the operations that can be performed in a stack follows.

Logical File Assignments. Assign decks are made up of one or more ASGN control cards specifying input/output devices that differ from the effective devices of the System Control Program. With the exception of the SYSTEM ASGN card, logical-file ASGN control cards can appear as frequently within the stack as the user wishes. Individual control cards within the deck can be in any order. The SYSTEM ASGN card appears once in a stack and immediately follows the Card Boot deck. A CORELOAD ASGN card is required if THRU EXECUTION is specified in a RUN card or if a

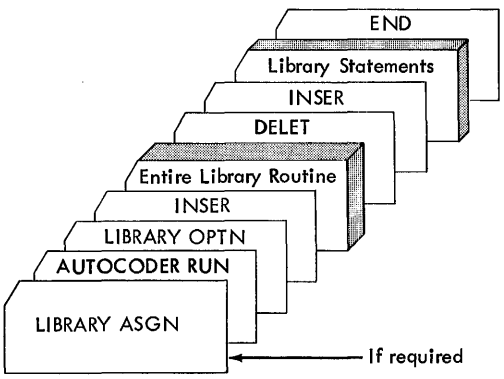

Figure 23. Library Change

CORELOAD OPTN card is included in the processor-job deck.

Library Maintenance. The composition of a library deck depends primarily upon the nature of the library job. However, an AUTOCODER RUN card is always required.

System Updating. Update decks as supplied by IBM are read by the System Control Program and must be available to the System on the device to which the CONTROL file is assigned. An update deck consists of one or more control cards, followed by any appropriate data cards.

Processor Runs. Runs depend upon a RUN card and the input to the processors. If the INPUT file is assigned to the same device as the CONTROL file (the card reader), each source deck must be placed behind its respective RUN control card. If the input to the processors is written in disk storage, an INPUT ASGN card is required designating the location of the source material in disk storage.

Communicating with the Operator. NOTE control cards and PAUSE control cards can appear anywhere in a stack between, not within, jobs. A HALT card must be the last card of a stack.

Preparing a Stack

The Card Boot deck, a SYSTEM ASGN card, and a HALT card are always required. The formats of the SYSTEM ASGN and HALT cards are shown in Appendix I.

The input cards for a stack are arranged in this order:

1. The 1402 or 1442 Card Boot deck.
2. The SYSTEM ASGN card.
3. Job decks, to include the assign card(s), library deck(s), update deck(s), and processor deck(s). Job decks can be in any order.
4. The HALT card.

This stack is placed in the card reader and is read by the Autocoder System.

Figure 24 shows a stack with CONTROL and INPUT files assigned to the same device.

Figure 25 shows a stack with CONTROL and INPUT files assigned to different devices.

Running a Stack

To perform a stack run when the System resides on 1311:

1. Place the System pack on the disk drive referred to in the SYSTEM ASGN control card, and ready the drive. (This card immediately follows the 1402 or 1442 Card Boot deck.)
2. Ready all the input/output devices to which the logical files are assigned. These are the assumed devices of the System Control Program and/or the devices defined by the ASGN cards. The assumed devices are: disk drive 0, the card reader, the card punch, and the printer.
3. Ready the console:
   a. Set the I/O check-stop switch off.
   b. Set the check-stop switch and disk-write switch on.
   c. Set the mode switch to RUN.
   d. Press CHECK RESET and START RESET.
4. Load the program.
   a. 1402 Card Reader: Press LOAD.
   b. 1442 Card Reader: Press START on the reader, and PROGRAM LOAD on the console.
5. When the System attempts to read the last card:
   a. 1402 Card Reader: Press START.
   b. 1442 Card Reader: Press START on the card reader.

To perform a stack run when the System resides on 1301:

1. Ready all the input/output devices to which the logical files are assigned. These are the assumed devices of the System Control Program and/or the devices referred to in the ASGN cards. The assumed devices are: disk unit 0, the card reader, the card punch, and the printer.
2. Ready the console:
   a. Set the I/O check-stop switch off.
   b. Set the check-stop switch and disk-write switch on.
   c. Set the mode switch to RUN.
   d. Press CHECK RESET and START RESET.
3. Load the program:
   a. 1402 Card Reader: Press LOAD.
   b. 1442 Card Reader: Press START on the reader, and PROGRAM LOAD on the console.
4. When the system attempts to read the last card:
   a. 1402 Card Reader: Press START.
   b. 1442 Card Reader: Press START on the card reader.

Loading Object Programs

Punched-card object programs can be executed independently of the Autocoder System. The procedures to be followed when a card-read error occurs depend on the format of the program and the object system.

To load the program:

1. Place the object deck in the card reader. (If for any reason the user does not wish to clear storage before

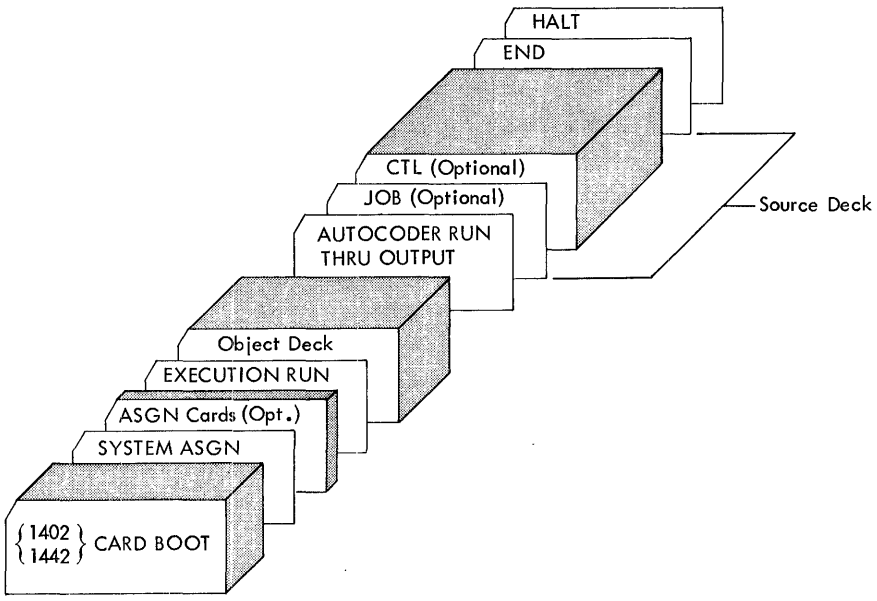

Figure 24. Stack with CONTROL and INPUT Files Assigned to the Same Device

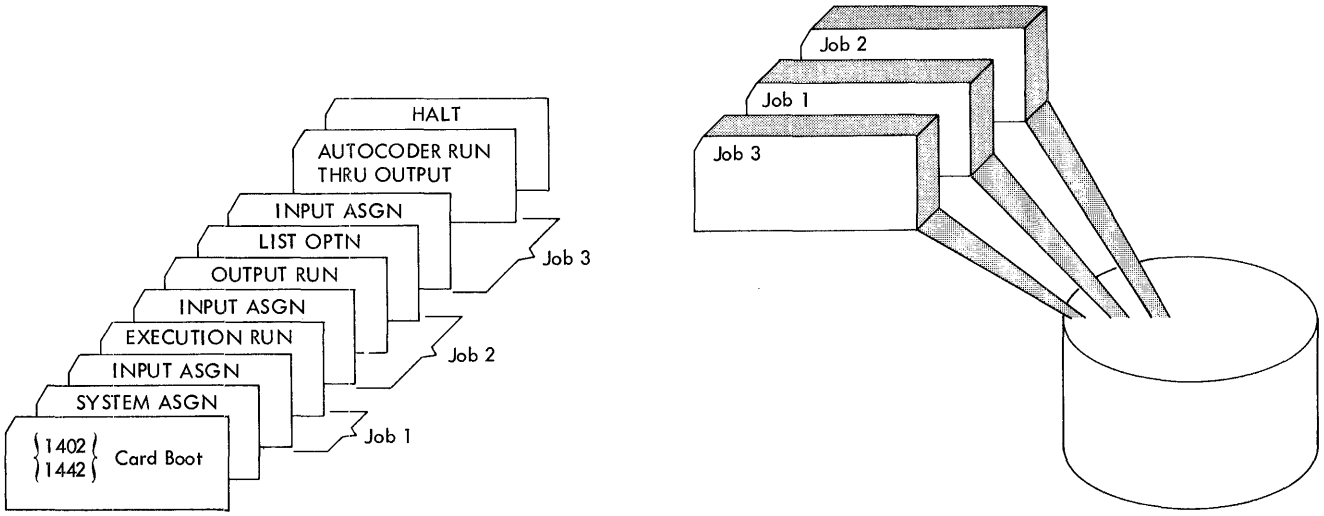

Figure 25. Stack with CONTROL and INPUT Files Assigned to Different Devices

loading the object program, he should remove the first two cards from the deck. These are the clear-storage cards generated by the processor.)

2. Set the I/O check-stop switch on. Set sense switches as needed by the object program.
3. Press CHECK RESET and START RESET.
4. Load the program:
   a. 1402 Card Reader: Press LOAD.
   b. 1442 Card Reader: Press START on the card reader, and PROGRAM LOAD on the console.
5. When the system attempts to read the last card:
   a. 1402 Card Reader: Press START.
   b. 1442 Card Reader: Press START on the card reader.

If a card-read error occurs while loading an object-program deck with the I/O check-stop switch on, the following procedures are followed to correct the error. If the reader is a 1402:

1. Non-process run-out the cards in the card reader.
2. Place the last three cards (two non-processed cards and the card in error) in the hopper.
3. Press CHECK RESET on the reader and START.

If the reader is a 1442 and the object-program deck is in the 1440 condensed-loader format:

1. Non-process run-out the cards in the card reader.
2. Place the last two cards in the hopper.
3. Press CHECK RESET and START RESET.
4. Set the I-address register to the ninth position of the loader.
5. Press START on the reader and START on the console.

If the reader is a 1442 and the object-program deck is in the 1440 self-loading format:

1. Non-process run-out the cards in the card reader.
2. Place the last two cards in the hopper.
3. Press CHECK RESET and START RESET.
4. Set the I-address register to 00073.
5. Press START on the reader and START on the console.

Halts and Messages

The halts and messages shown in Figure 26 can appear during a stack run. To display halt numbers, press the A-address register key. Messages are printed on the MESSAGE file.

Halt Number (A-Address Register)	MESSAGE and/or Meaning	Restart Procedure
1bb	Card read error.	1. 1402 card reader: non-process run-out the cards in the reader. Place the last three cards (two non-processed cards and the card in error) in the hopper. Press START. 2. 1442 card reader: non-process run-out the cards in the reader. Place the two non-processed cards in the hopper. (The first non-processed card is the card in error). Press START on the reader and START on the console.
2bb	Wrong-length record or no-address-compare error sensed ten times during a disk-read or write operation.	Press START for ten disk-read or write retries.
3bb	Parity error sensed ten times during a disk-read or write operation.	Press START for ten disk-read or write retries.
4bb	Not-ready condition sensed when a disk-read or write operation was attempted.	Ready the disk unit and press START.
5bb	1. Librarian-control OPTN card is incorrect, or 2. Preprocessor phase not on the SYSTEM file.	1. Non-process run-out the cards in the card reader, correct the OPTN card, and restart the system, or 2. If the OPTN card is not incorrect, use the part of the system deck labeled AUTOCODER PREPROCESSOR and rebuild the preprocessor portion of the system. Follow the procedures as described in Building an Autocoder System.
6bb	One of the following messages precedes this halt: ERROR HEADER ABOVE UNKNOWN A phase update card specifies a phase name that is not in the phase table. ERROR NO KNOWN TYPE OF UPDAT Columns 21- ? of a phase update card are incorrect. ERROR CYLINDER OVERFLOW The phase update card specifies that the phase is to be placed on a set of sectors that exceeds one cylinder. ERROR ACTUAL IDENT UNEQUAL TO HEADER IDENT Columns 76-80 of a change card do not contain the phase name specified in columns 6-10 of the update control card associated with it. ERROR NON CONTROL CARD WITHOUT CONTROL PRECEDING An update card is missing, out of sequence, or mis-punched. ERROR UNKNOWN EXECUTE CARD A change card with 006 punched in columns 1-3 does not have =, or =/ or =M punched in columns 6 and 7. These punches are found in set-word-mark or clear cards developed for a DA statement. No other types of special execute cards are permitted. ERROR PATCH ABOVE OUTSIDE OF PROGRAM LIMITS The phase area cannot contain the data specified in the change cards.	The contents of the error cards are printed. Non-process run-out the cards in the card reader, correct the error card, and restart the update operation. Corrections successfully completed before the halt occurs need not be reprocessed.

Figure 26. Halts and Messages (Part 1 of 5)

Halt Number (A-Address Register)	MESSAGE and/or Meaning	Restart Procedure
	ERROR CHARACTER COUNT TOO LARGE A change card contains a character count greater than 67 characters. The character count is punched in columns 4 and 5. ERROR ABOVE CARD CREATES GROUP MARK WORD MARK A set-word-mark card developed for a DA statement attempts to set a word mark over a position containing a group mark, or a condensed card contains a word separator character followed by a group mark. This is an error because a group mark with a word mark can neither be read from nor written in disk storage.	
7bb	More than 50 different DTF entries used in the program.	Correct the source program and reassemble the source program from the beginning.
8bb	CONTROL CARD ERROR LIBRARY OPTN This halt indicates one of the following conditions: 1. An INSER, DELET, or END card is missing or mispunched. 2. An attempt to insert or delete entries in a library routine that does not exist. 3. Entries not in collating sequence, according to macro name and/or sequence number.	The contents of the incorrect card(s) are printed. Remove the incorrect card(s) and place the remainder of the cards in the card reader. If the library change operation is not completed, the LIBRARY file cannot be used.
9bb	Any disk error that occurs while the bootback routine is returning control to the System Control Program.	Press START for one disk retry.
10bb	More than 300 macros within macros have been used in the source program.	Correct the source program and reassemble the source program from the beginning.
11bb	WORK1 capacity exceeded during an AUTOCODER RUN THRU OUTPUT or an AUTOCODER RUN THRU EXECUTION, or OUTPUT-file capacity exceeded during an AUTOCODER RUN.	Change the WORK1 or OUTPUT ASGN card and restart the assembly of the job.
12bb	Disk-error condition sensed during the Preprocessor phase.	Press START for ten disk retries.
13bb	LIBRARY file capacity exceeded. Part of the library routine that was being processed when the halt occurred will be in the LIBRARY file. All library routines following the routine being processed will no longer be in the LIBRARY file.	To finish the job: 1. 1402 card reader: nonprocess run out the cards in the reader. Place the END card in the hopper. Press START. 2. 1442 card reader: nonprocess run out the cards in the reader. Place the END card in the hopper. Press START on the reader and START on the console. To determine the names of the routines remaining in the LIBRARY file, perform a library-listing operation and specify HEADERS in the LISTING OPTN card.
14bb	Invalid card encountered during LIBRARY OPTN run.	Press START to complete the run. The invalid card will be bypassed. If the run is not completed, the LIBRARY file cannot be used.
22bb	More than 30 different INCLD routines used in one overlay.	Correct the source program and reassemble the source program from the beginning.
33bb	Library table (99 macro names) exceeded.	To finish the job: 1. 1402 card reader: nonprocess run out the cards in the reader. Place the END card in the hopper. Press START. 2. 1442 card reader: nonprocess run out the cards in the reader. Place the END card in the hopper. Press START on the reader and START on the console. To determine the names of the routines in the LIBRARY file, perform a library-listing operation and specify HEADERS in the LISTING OPTN card.

Figure 26. Halts and Messages (Part 2 of 5)

Halt Number (A-Address Register)	MESSAGE and/or Meaning	Restart Procedure
001	WRONG SYSTEM The message appears unconditionally on the printer.	1. Non-process run-out the cards in the reader. 2. Correct the SYSTEM ASGN card, or place the correct pack on the unit indicated in the SYSTEM ASGN card. 3. Restart the stack.
002	TEN RD TRIES PRESS STRT FOR 10 MORE The message appears unconditionally on the printer. It indicates any disk error while attempting to read the SYSTEM file.	Press START for ten disk-read retries.
003	SYSTEM ASGN NOT SENSED The SYSTEM ASGN card did not immediately follow the Card Boot.	1. Non-process run-out the cards in the reader. 2. Place the SYSTEM ASGN card and the remainder of the stack in the read hopper. 3. If the reader is 1402, press START. 4. If the reader is 1442, press START on the reader and START on the console.
004	Parity check, wrong-length record, or no-address-compare error sensed 10 successive ti nes during disk bootstrap operation.	Press START for 10 disk-read retries.
005	End-of-file sensed in SYSTEM file during disk bootstrap operation.	Non-process run-out the cards in the reader and restart the stack.
006	HALT card image Indicates the end of the stack.	Hard halt.
007	Card-punch error.	1. 1402 card punch and 1444 card punch: non-process run-out the cards in the punch. Discard the last three cards (two non-processed cards and the card in error) in the stacker. Press START. 2. 1442 card punch: discard the last card in the stacker. Press START on the punch and START on the console.
008	Card-read error.	1. 1402 card reader: non-process run-out the cards in the reader. Place the last three cards (two non-processed cards and the card in error) in the hopper. Press START. 2. 1442 card reader: non-process run-out the cards in the reader. Place the two non-processed cards in the hopper. Press START on the reader and START on the console.
009	Printer error.	1. 1403 printer: press START. 2. 1443 printer: press START on the printer and START on the console.
010	Non-blank card at the punch station in the 1442 card read-punch.	Non-process run-out the cards in the 1442. Place blank cards before the non-processed cards. Press START on the 1442 and START on the console.
011	PAUSE card image.	Press START.
012	Console-printer error	Press START for one retry of the read or write operation.
013	***ASGN card image The halt indicates that the ASGN card is incorrectly punched.	1. 1402 card reader: the card in the stacker is the incorrect ASGN card. Correct the ASGN card. Non-process run-out cards in the reader. Place the corrected ASGN card and the two non-processed cards in the hopper. Press START. 2. 1442 card reader: non-process run-out the cards in the reader. The first non-processed card is the incorrect ASGN card. Correct the ASGN card. Place the corrected ASGN card and the second non-processed card in the hopper. Press START on the reader and START on the console.

Figure 26. Halts and Messages (Part 3 of 5)

Halt Number (A-Address Register)	MESSAGE and/or Meaning	Restart Procedure
		3. If the user wishes, he can ignore the two steps outlined above, and press START. The system will then use the effective device assignment for that particular file.
040	The logical file has been assigned to an area that overlaps a previously defined file label. (1311 only.)	Hard halt. Change the assignment and restart the stack with the Card Boot.
168	Phase not found in phase table while in supervisory call for phase.	A part of the System must be rebuilt. Use the parts of the System deck labeled CARD BUILD, SYSTEM CONTROL, and AUTOCODER PROCESSOR. Follow the procedures as described in Building an Autocoder System.
500	Disk not ready.	Ready the disk unit and press START.
629	Parity check, wrong-length record, or no-address-compare error sensed 10 successive times during a disk-read or write operation.	Press START for 10 disk-read or write retries.
777	This halt will occur if the work areas are not large enough.	Hard Halt. Enlarge work areas to required size and restart the assembly.
1250	END OF CONTROL CARD DIAGNOSTICS NOTE — PRESS START TO ASSEMBLE, START-RESET AND START TO BYPASS ASSEMBLY	As indicated in the message.
1447	NOTE — ASSEMBLY ERRORS — PRESS START TO EXECUTE, START — RESET AND START TO BYPASS EXECUTION	As indicated in the message.
1833	NOTE — DIAGNOSTICS — PRESS START TO ASSEMBLE, START — RESET AND START TO BYPASS ASSEMBLY	As indicated in the message.
	START ADDRESS OF INPUT FILE DOES NOT REFER TO HEADER RECORD	If a message is printed and no halt occurs, the next control card is processed.
	EXPECTED HEADER # (52 positions) #, FOUND / (52 positions) /	
	EXPECTED ID #XXXXX#, FOUND /XXXXX/	
	NOTE card image	
	*** card image All cards not recognized by the System Control Program are flagged (***), written on the MESSAGE file, and bypassed by the System.	
	Card image INVALID UPDAT TYPE Update card with invalid update mode designated.	
	PHASE XXX ALREADY ON SYSTEM. WILL DROP THIS SET OF CARDS	
	PHASE XXX NOT FOUND	
	HEADER CARD ERROR All header cards must have 24232 in columns 1 through 5.	
	Card image PHASE AREA EXCEEDED	
	****PROCESSOR UNKNOWN****	

Figure 26. Halts and Messages (Part 4 of 5)

Halt Number (A-Address Register)	MESSAGE and/or Meaning	Restart Procedure
	CORELOAD NOT ASSIGNED, OPTION NOT DONE The next output option is processed.	
	CORELOAD FILE NOT ASSIGNED, OPTION NOT DONE AND EXECUTION SUPPRESSED	
	Image of an output option card - OPTION UNKNOWN The next output option card is processed.	
	CORELOAD HEADER — (52 positions), ID — (5 positions) Use the information in an EXECUTION RUN card.	
	CORELOAD OUTPUT COMPLETE ON {1311 / 1301} UNIT n, START nnnnnn, END nnnnnn The START address is address of the object program header record. The END address is the address of the next available sector. Use the information in an INPUT ASGN card for an EXECUTION RUN.	
	{LST / OUT / INP} FILE {STARTS / ENDS} ON {1311 / 1301} UNIT n AT ADDRESS nnnnnn	
	XXXXX MACRO NOT IN LIBRARY The macro requested (XXXXX) is not in the LIBRARY file.	
	END OF LISTING OPTN The library-listing job has been completed.	
	XXXXX BLOCKS LEFT EOJ The library-change operation has been completed. XXXXX is the number of blocks available in the LIBRARY file.	
	END OF SYSTEM OPTN The update operation has been successfully completed.	
	LIBRARY FILE NOT RECOGNIZED The library has not been assigned correctly or the library has not been initialized.	
	OUTPUT FILE NOT ASSIGNED TO DISK The RUN card specifies AUTOCODER RUN. The Autocoder text must be written on disk.	
	INPUT FILE NOT ASSIGNED TO DISK The RUN card specifies OUTPUT RUN or OUTPUT RUN THRU EXECUTION. An INPUT ASGN card, designating the location of the Autocoder Text, is required.	
	NO TEXT IN INPUT FILE 1EOFb sensed in the INPUT file and the Autocoder Text has not been processed; or, the assigned INPUT file does not contain text.	

Figure 26. Halts and Messages (Part 5 of 5)

Methods of Execution

An object program can be executed by using any of the following methods:

1. Load-and-go
2. Delayed execution
   a. Controlled
   b. Independent.

Load-and-Go

Automatic loading and execution of an object program is a processor job. The object program is written on the CORELOAD file, loaded into core storage, and control is transferred to it. See *Autocoder Run Thru Execution* and *Output Run Thru Execution* for the formats of the required control cards.

Delayed Execution

Execution of an object program that has been produced by an AUTOCODER RUN THRU OUTPUT or an OUTPUT RUN is called *delayed execution.* The object program can be executed under control of the Autocoder System or independently of the System.

Controlled Execution

Controlled execution is initiated by an EXECUTION RUN card. See *Delayed Execution* under *Preparing Processor Jobs* for the formats of required control cards.

The advantages gained by executing programs under control of the Autocoder System are:

1. Object programs, in the coreload format or punched into cards, can be selected and executed.
2. More than one object program in the coreload format can be loaded from disk storage and executed during an EXECUTION RUN. (See *Multi-Program Execution.*)
3. Control can be returned to the System Control Program after object-program execution. This feature makes it possible to perform the next job without operator intervention.

Processor Action

If the input is in the coreload format, the Execution processor:

1. Determines if the START address of the INPUT file refers to a header record. If the record is not a header record, a message is printed on the MESSAGE file, and control reverts to the System Control Program.
2. Compares columns 21-72 and 76-80 of the EXECUTION RUN card, if punched, with the portion of the header record that contains JOB card information. If the fields are not identical, a message is printed on the MESSAGE file and control reverts to the System Control Program. The message consists of the information that is supposed to be in the record and the information that is in the record.
3. Clears storage from the disk loader down to position 1.
4. Transfers program control to the disk loader. After the loading process has been completed, control is transferred to the object program.

If the input is from cards, the object-program clear-storage cards are bypassed and storage is cleared from the bootstrap routine down to location 80. Control is then transferred to the object-program bootstrap routine.

If the loader has not been destroyed during a program overlay, loading can be resumed by branching to the disk loader (3701 for 4K, 7701 for 8K, 11701 for 12K, and 15701 for 16K). The inclusion of the LDRCL macro in the source program provides the linkage.

Multi-Program Execution

More than one object program, in the coreload format, can be loaded and executed during an EXECUTION RUN. If the object programs are to be selected randomly, the last two instructions in all object programs, except the last object program to be executed, must be:

1. An instruction that moves the disk unit number and the disk address of the *next* object program's header record into the core storage locations that contain the address of the next sector to be read. (3825-3831 for 4K, 7825-7831 for 8K, 11825-11831 for 12K, and 15825-15831 for 16K.) Thus, the *next* object program must have been assembled and transferred to the CORELOAD file before the instruction can be coded. The message that is printed after an object program has been transferred to the CORELOAD file gives the address of the object program's header record.
2. A branch to the disk loader. (3701 for 4K, 7701 for 8K, 11701 for 12K, and 15701 for 16K.) The inclusion of the LDRCL macro instruction as the last in-

struction before the source program END statement will provide the linkage to the disk loader.

If the object programs are batched on the CORELOAD file and are to be processed sequentially, only the branch to the disk loader is required. Because the object programs are stored sequentially, the address of the *next* object program's header record is in the core storage locations that contain the address of the next sector to be read.

After executing an object program, the disk loader does not clear core storage. The user should clear core storage (if required) before calling the next object program for execution.

Note: The disk loader, which begins at 3701 for 4K, 7701 for 8K, 11701 for 12K, and 15701 for 16K, must not be destroyed during program execution. The loader itself requires 134 positions. The remaining high core-storage positions contain the read-in area for the loader and the bootback routine.

Example: Assume that two object programs are to be selected randomly during a single EXECUTION RUN and that the object system size is 12K. The procedure would be:

1. Assemble the second object program to be executed by performing an AUTOCODER RUN THRU OUTPUT with the CORELOAD OPTN specified.
2. Record the message. Assume that the message is CORELOAD OUTPUT COMPLETE ON 1311 UNIT 1, START 012300, END 012399.
3. Code the last three source statements in the first program to be executed as shown in Figure 27. Assume that the symbolic address of the first instruction to be executed is BEGIN. In a 12K machine, positions 11825-11831 contain the address of the next sector to be read.
4. Assemble the first object program to be executed by performing an AUTOCODER RUN THRU OUTPUT with the CORELOAD OPTN specified.
5. Record the message. Assume that the message is CORELOAD OUTPUT COMPLETE ON 1311 UNIT 1, START 004596, END 004960.
6. When the EXECUTION RUN is to be performed, use:
   a. An ASGN card that specifies INPUT ASGN 1311 UNIT 1, START 004596, END 004960. This ASGN card assigns the INPUT file to the disk area that contains the first object program to be executed. Because the second program is not selected for loading by the Execution processor, the area assigned to the INPUT file need not contain the second program.
   b. An EXECUTION RUN card whose operand contains the information from the last JOB card in the first program to be executed.

Label	Operation	OPERAND
	MLC	@10123000@,11831
	LDRCL	
	END	BEGIN

Figure 27. Coding for Multi-Program Execution

Example. Assume that three object programs are to be stored and executed sequentially. The procedure would be:

1. Assemble the programs. Perform a series of AUTOCODER RUN THRU OUTPUT jobs with the CORELOAD OPTN specified for each job. (The LDRCL macro instruction must be the last instruction of the first two source programs.)
2. Record the messages that are printed after the coreload operations are completed.
3. When the EXECUTION RUN is to be performed, use:
   a. An INPUT ASGN card that specifies the disk area that contains the programs. The START address must be the address of the first object program's header record. Use the START address from the first message and the END address from the last message.
   b. An EXECUTION RUN card whose operand contains the JOB card information from the last JOB card in the first program to be executed.

Return to System Control

If control is to be transferred to the System Control Program after an object-program execution, the object program must contain an instruction to branch to the bootback routine (3928 for 4K, 7928 for 8K, 11928 for 12K, and 15928 for 16K). The inclusion of the SYSCL macro in the source program provides the linkage.

Independent Execution

Any object-program deck produced by the Autocoder System can be executed independently of the System. The user need not prepare any additional cards to initialize execution of the program.

Condensed-Loader Considerations

The factors to be considered when using the condensed-loader format are:

IBM 1440

Read-In Area. The read-in area is assumed to extend from 0001 to 0074. A group-mark word-mark is located in the last position of the read-in area.

The read-in area may be relocated by specifying the starting address in the source program CTL card. Since the loader uses a clear-storage command to clear the read-in area from the end address, the beginning and ending addresses must have the same hundreds position (Example: 00901-00974).

Loader Instruction Area. The loader instruction area is assumed to be at 0075-0206 (132 characters). In the special case where the card-input area starts in

an address greater than 999 and the modify address feature is not available, the loader instruction area is extended to 145 characters.

The loader instruction area can be relocated by specifying the starting address of the loader in the source program CTL card. The entry point to the loader, after the execution of a program overlay, is 0083 for the assumed location and start address +8 positions for the relocated loader.

IBM 1401 or 1460

Read-In Area. The read-in area is fixed at location 0001-0080.

Loader Instruction Area. The loader instruction area is assumed to be at 0081-0205 with a length of 125 characters. This area may be relocated by specifying the starting address in the source program CTL card. The entry point to the loader after the execution of a program overlay is 0081 for the assumed location and start address for the relocated loader.

Revising the Object Program

It is possible to correct or revise an object program deck without reassembling. Either of two methods can be used:

1. The user duplicates the condensed card, substituting the correct information where needed. The corrected card is then placed in the proper location within the condensed deck before loading the object program. To determine the proper location, check the program listing. The sequence numbers punched in columns 72-75 of the program deck correspond to the sequence numbers in the program listing.
2. The correct information is loaded into storage after the original object program has been loaded, overlaying part of the original object program. The user punches patch card(s) and places them just before the assembled END, XFR, or EX card in the object program or program segment to which the patch applies. (Check the listing for number of the END, XFR, or EX assembled instruction.)

Condensed-Loader Format

A patch card for an object program in the condensed-loader format is punched as follows:

Columns	*Contents*
1-3	The three-character machine address of the first storage position to be loaded.
4-5	The number of characters to be loaded from the card. Word-separator characters are not counted.
6-71	The instructions and/or constants to be loaded. A word-separator character (0-5-8 punch) precedes every character requiring a word mark in core storage.
72-75	The source-program sequence number of the first instruction or constant in the card.
76-80	The program identification.

Punching in columns 72-80 is not required for loading. Thus, the card sequence number and the program identification fields may be left blank. Note that if two successive card columns both contain word-separator characters, one word-separator character will be loaded into core storage without a word mark.

Self-Loading Format

A patch card is punched in the following format:

Columns	*Contents*
1-39	The constant(s) or machine-language instruction(s) to be loaded into storage. The information must be left-justified in this field.
40-46	A load instruction that loads the above data into storage with a high-order word mark.
47-53	If the data should not have a high-order word mark, this field contains a seven-character clear-word-mark instruction. If the high-order word mark is to be left in storage, this field contains: 1. A set-word-mark instruction. If two or more instructions have been loaded into core storage, a word mark must be set for each instruction, or: 2. A NOP instruction (N000000), if additional word marks are not needed.
54-60 61-67	These fields contain set-word-mark or NOP instructions (see preceding paragraph).
68-71	For the 1401-1460: 1040. An instruction to read a card and branch to location 040, which is the address of the next load instruction or an execute instruction. For the 1440: B073. An instruction to branch to the read-a-card and branch instructions which are in positions 73-85.
72-75	Program-listing sequence number of the first instruction and/or constant in the card.
76-80	Program identification.

Punching in columns 72-80 is not required for loading. Thus, the sequence number and program identification field may be left blank.

Autocoder-System Deck Description and Preparation

The System deck supplied to the user contains six sections as shown in Figure 28. One section, Marking Program, is used to separate the sections for ease in labeling the various components of the complete deck. Three sections, Write File-Protected Addresses, System Control Card Build, and Autocoder Update are used to build the System. A fifth section, the Card Boot, is used to operate the System. A sixth section, Sample Program, is used to test the System built by the user. The individual sections are separated by the Marking Program control cards. In the instances where more than one set of cards makes up a section, a Marking Program control card separates the sets.

To facilitate building and maintenance operations, mark the sections as indicated by the Marking Program messages.

All cards in the System Deck, except the four 1402 load-card sets, the four 1442 load-card sets, the Autocoder macros, and the Autocoder Sample Programs, contain a sequence number in columns 72-75. The cards are numbered consecutively, beginning with 0001.

All load cards contain a sequence number in column 80. Each set of 1402 load cards is numbered consecutively from 1 through 6 and is identified by a 0-4-8 punch (% symbol) in column 79. Each set of 1442 load cards is numbered consecutively from 1 through 7 and is identified by a 3-8 punch (# symbol) in column 79.

Each Autocoder macro contains a sequence number in columns 1-5, beginning with either 00000 or 00010. Each Autocoder Sample Program contains a sequence number in columns 1-5, beginning with 0001b.

If it is necessary to resequence the System deck, the user should sort the cards in the following manner:

1. Sort on column 79 (0-4-8 punch) to select the 1402 load cards.
2. Sort the 1402 load cards on column 80 to sequence the cards.
3. Assemble the four sets of 1402 load cards.
4. Sort on column 79 (3-8 punch) to select the 1442 load cards.
5. Sort the 1442 load cards on column 80 to sequence the cards.
6. Assemble the four sets of 1442 load cards.
7. Sort the remainder of the System deck on columns 75, 74, 73, and 72. After sorting, the Autocoder Macros and Sample Programs will be in the reject pocket.
8. Check the program listing, which is supplied with the System deck, and insert the sets of load cards in the appropriate places.
9. Sort the Autocoder Macros and Sample Programs on column 5. After sorting, the Sample Programs will be in the reject pocket.
10. Sort the Autocoder Macros on columns 4, 3, 2, 1, 80, 79, 78, 77, and 76.
11. Check the program listing and insert the Autocoder Macros.
12. Sort the Sample Programs on columns 4, 3, 2, 1, and 80.
13. Check the program listing and insert the Sample Programs.

Marking Program

The Marking Program deck is made up of two sets. The set for the 1442 consists of 13 cards and has identification code 50zy1 punched in columns 76-80. The set for the 1402 consists of 11 cards and has the identification code 50zz1 punched in columns 76-80. A blank (except for sequencing in columns 72-75) card follows each set.

The Marking Program separates the various sections and sets that make up the System deck. When a control card is sensed, a halt occurs and a message is printed.

If the reader is 1442, the initial message is:
HALT AT EACH DECK SEGMENT. DISCARD FIRST CARD, MARK DECK AS PRINTED, PRESS START TO CONTINUE.

If the reader is 1402, the initial message is:
HALT AT EACH DECK SEGMENT. MARK DECK AS PRINTED, PRESS START TO CONTINUE.

Subsequent messages contain the name of the section to be marked.

To use the decks:

1. Set sense switch A on. Set all other sense switches off.
2. Set the I/O check stop switch off.
3. Press CHECK RESET and START RESET.
4. Select the Program Marking deck that is appropriate for the system and remove the other deck.
5. Remove the blank card following the Marking Program and place the program in the card reader, followed by the remainder of the Autocoder System deck.
6. Load the program.
   a. 1402 Card Reader: Press LOAD.
   b. 1442 Card Reader: Press START on the reader, and PROGRAM LOAD on the console.

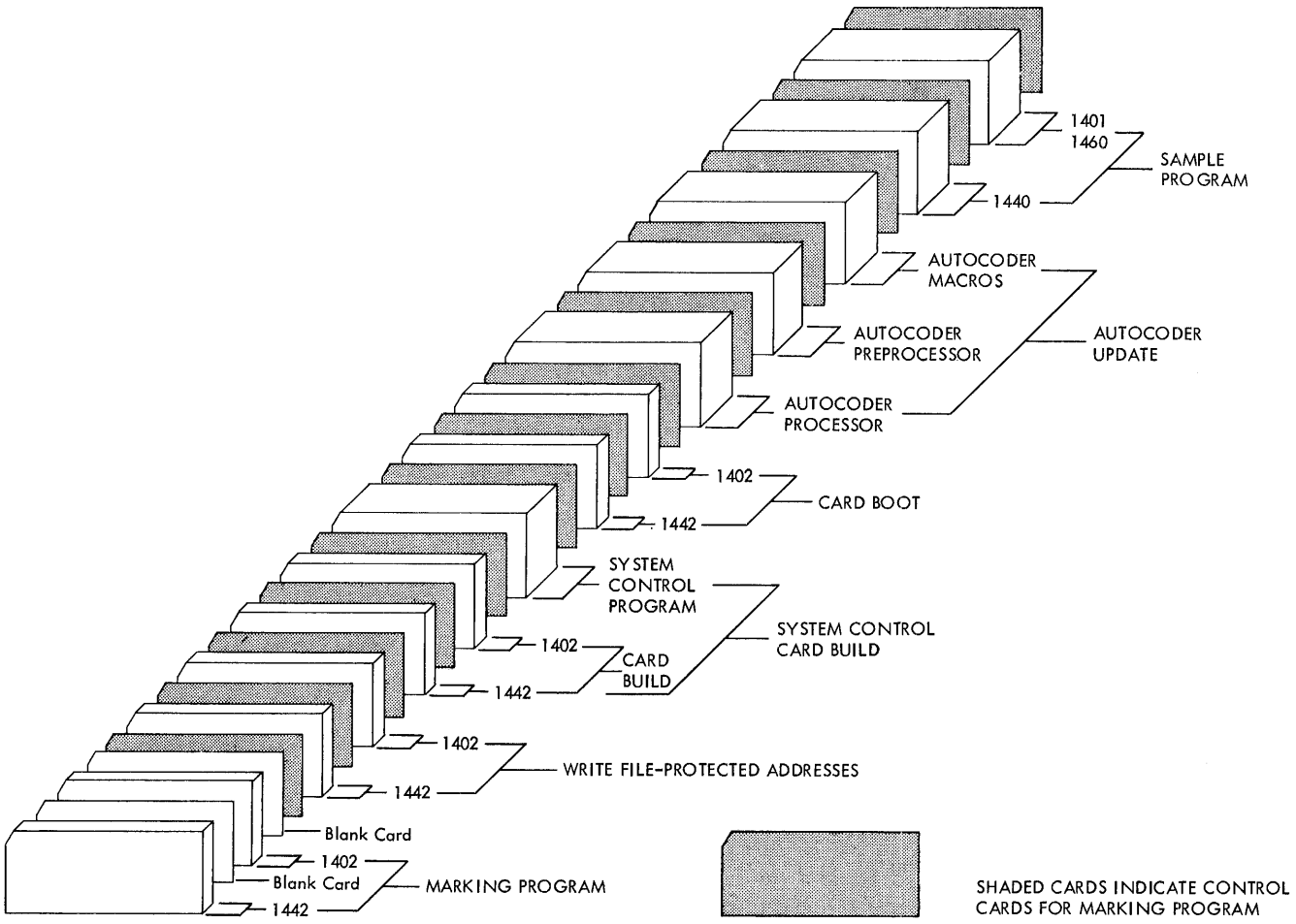

Figure 28. Autocoder System Program Deck

7. Halt 003 procedure.
   a. 1402 Card Reader: Press START. The Marking Program is in the NR stacker.
   b. 1442 Card Reader: Remove the Marking Program from stacker 1 and press START on the console.
8. Halt 001 procedure.
   a. 1402 Card Reader: Remove the cards from stacker 1 and press START. Mark the deck section as indicated in the message. The Marking Program control card is in the NR stacker.
   b. 1442 Card Reader: Remove the cards from stacker 1 and press START on the console. Discard the first card (Marking Program control card) and mark the section as indicated in the message.

   Note: The Marking Program control cards are identified by ##### in columns 1-5. These cards are only for the use of the Marking Program and should be discarded after the deck is marked.

9. When the system attempts to read the last card.
   a. 1402 Card Reader: Press START.
   b. 1442 Card Reader: Press START on the reader. The last card is a Marking Program control card and should be discarded.

The following halts can occur when using the Marking Program. To display the halt number, press the A-address register key.

Halt Number A-Address Register	*Meaning*
001	The deck section in stacker 1 should be marked.
002	End of job.
003	The initial message has been printed.
008	Card-read error. To retry the operation, *For the 1402:* Non-process run-out the cards. Remove the last three cards in the stacker and place them in the hopper. Press START. *For the 1442:* Non-process run-out the cards. Place the two non-processed cards in the read hopper. Press START on the reader and START on the console.
009	Printer error. To retry the operation, a. 1403 Printer: Press START. b. 1443 Printer: Press START on the printer and START on the console.

Write File-Protected Addresses

The Write File-Protected Addresses section is punched in the Autocoder condensed-loader format. The deck consists of approximately 120 cards.

The set of cards for the 1442 has the identification code 50FS1 punched in columns 76-80. The set of cards for the 1402 has the identification code 50FP1 punched in columns 76-80.

This section writes disk addresses whose values are equal to the normal addresses plus 260,000. It is by use of these false addresses that the file-protected area is created.

System Control Card Build

This section contains control cards and cards punched in the Autocoder condensed-loader format. It includes both the 1402 and the 1442 Card Build programs and the System Control Program. All necessary control cards are incorporated within the section, which consists of approximately 1000 cards.

The Card Build set for the 1442 has the identification code 50X41 punched in columns 76-80. The Card Build set for the 1402 has the identification code 50X01 punched in columns 76-80.

The System Control Program section is identified by the code 50SX1 punched in columns 76-80, where *x* is alphameric. The section loads the System Control Program in disk storage.

Card Boot

The 1402 Card Boot set, consisting of 17 cards, and the 1442 Card Boot set, consisting of 19 cards, are punched in the Autocoder condensed-loader format. The 1442 Card Boot set has the identification code 50SZ1 punched in columns 76-80. The 1402 Card Boot set has the identification code 50PZ1 punched in columns 76-80.

Because the Card Boot is required for each stack of jobs to be performed by the System, the Card Boot must be removed and saved for future System operations.

Autocoder Update

The Autocoder Update section is made up of three sets of cards: Autocoder Processor, Autocoder Preprocessor, and Autocoder Macros. There are approximately 2900 cards in the entire section. The Autocoder Processor set is in the Autocoder condensed loader and the UPDAT card formats. There are approximately 1400 cards in the Autocoder Processor set. Approximately 800 cards are identified by the code AUXXA punched in columns 76-80. Approximately 500 cards are identified by OPXOX in columns 76-80. Approximately 100 cards are identified by EXXXX punched in columns 76-80. In each case, *x* is alphameric.

The Autocoder Preprocessor set contains the fixed phases that are not under the direct control of the System Control Program. These phases reside outside the file-protected area of the Autocoder System. An AUTOCODER RUN card precedes the set and contains blanks in columns 76-80. The set contains approximately 750 cards, each of which is identified by an alphameric phase name in columns 76-80. (See Appendix II for a list of Autocoder Preprocessor phase names and functions.)

The Autocoder Macro set is punched in the Autocoder library format. All required control cards are incorporated within the set. An AUTOCODER RUN card and a LIBRARY OPTN card precede the set. Each of these two cards contains blanks in columns 76-80. The set contains approximately 750 cards, each of which is identified by a macro name in columns 76-80.

The macros included in this set are CALL, MA (modify address), LOOP, COMPR (compare), LDRCL and SYSCL (linkage), ADD, SUB, MLTPY, and DIVID.

Note: The CHAIN and INCLD macros are incorporated in the Autocoder Preprocessor.

Sample Program

The Sample Program section is punched in the Autocoder source language and contains approximately 100 cards.

The 1442 set has the identification code SPL40 punched in columns 76-80. The 1402 set has the identification code SPL01 punched in columns 76-80.

Building an Autocoder System

After all sets of cards have been labeled and those sets of cards not applicable to the user's system have been removed, the user is ready to use the prepared System deck to build the Autocoder System.

Figure 29 is a block diagram showing the building of a disk-resident System.

The System unit must be prepared for writing the complete System from cards. The user must clear disk unit 0 in the move mode from 000000 to 000199, in the load mode from 000200 to 000259, in the move mode from 000260 to 000299, in the load mode from 000300 to 004799, and in the move mode from 004800 to 019979. The Clear Disk Storage program applicable to the user's system can be used for this operation.

Figure 30 shows the disk storage allocation on the System unit.

The control cards for the utility program must be punched in the following manner:

For 1311,

Columns	*Contents*
1-15	M00000000019900
21-35	L00020000025900
41-55	M00026000029900

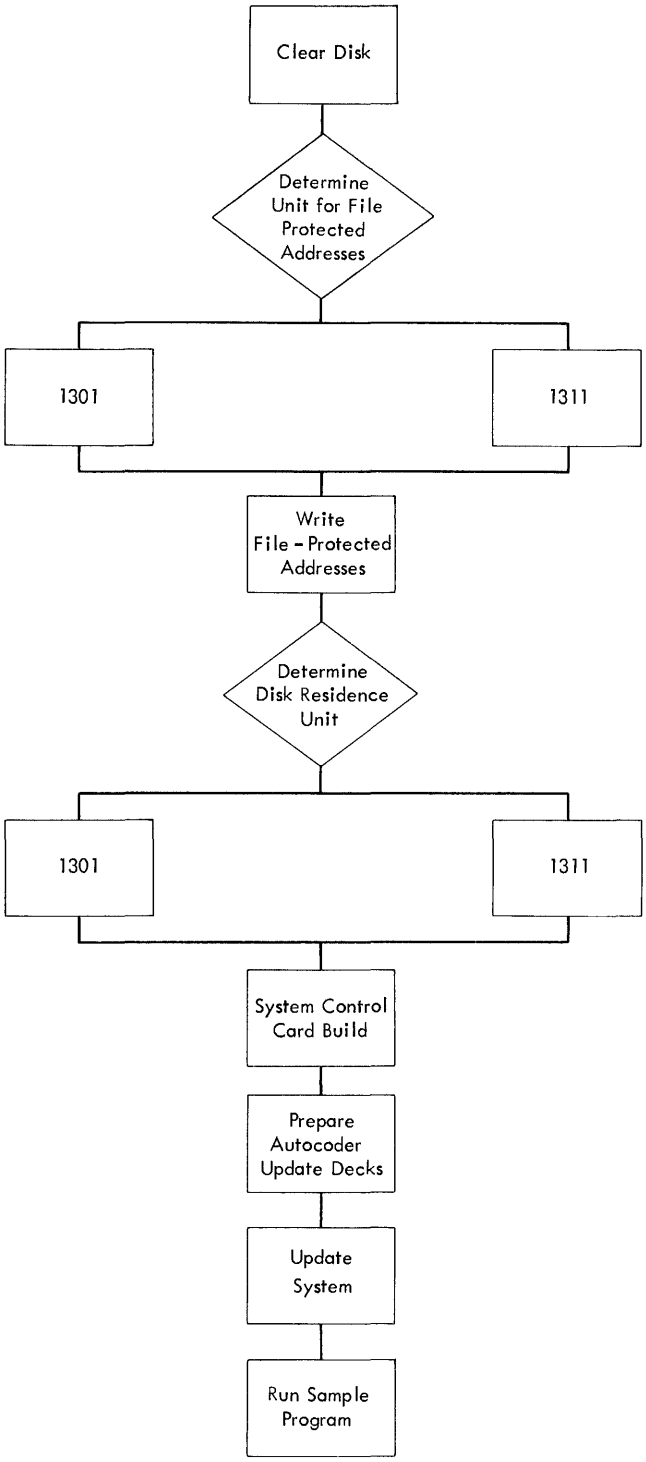

Figure 29. Building the System

Columns	*Contents*
1-15	L00030000479900
21-35	M00480001997900

For 1301,

The format of the control cards is the same as that given for 1311 except columns 14-15, 34-35, and 54-55 must contain record marks (0-2-8 punch) instead of zeros.

FILE	MODE	File-Protected	Sector Range
SYSTEM File			
Preprocessor	Move	No	000000-000199
	Load	No	000200-000259
	Move	No	000260-000299
	Load	No	000300-002499
System Control and Processors	Load	Yes	002500-004799
WORK1 File	Move	No	004800-011199
WORK2 File	Move	No	011200-012399
WORK3 File	Move	No	012400-012899
LIBRARY File	Move	No	012900-019979

Figure 30. Disk Storage Allocation

Write File-Protected Addresses

The last card in the section labeled WRITE FILE PROTECT is a control card that is partially prepunched. It is by the use of this control card that the limits of the file-protected area in the disk-storage unit are supplied. The user must indicate in the control card whether the System is to reside on a 1301 or 1311 disk unit. For both the 1301 and 1311, the System must be built on drive unit 0. In the case of the 1311, the System pack can be used on any drive once the System has been built. The control card is punched as follows:

Columns	*Contents*
1-15	FILE-PROTECT ON (prepunched)
17-20	1301 or 1311
22	0 (prepunched)
24-42	FROM NORMAL ADDRESS (prepunched)
44-49	002500 (prepunched)
51-52	TO (prepunched)
54-59	004800 (prepunched)

After columns 17-20 have been punched by the user, the card must be replaced as the last card of the section.

To use the section when the System is to reside on 1311:

1. Ready the pack on disk drive 0.
2. Set the write-address mode switch on.
3. Set the write-disk switch on.
4. Set the I/O check stop switch on.
5. Press CHECK RESET and START RESET.
6. Place the Write File-Protected Addresses section in the card reader.

7. Load the program.
   a. 1402 Card Reader: Press LOAD.
   b. 1442 Card Reader: Press START on the reader, and PROGRAM LOAD on the console.
8. When the system attempts to read the last card,
   a. 1402 Card Reader: Press START.
   b. 1442 Card Reader: Press START on the reader.
9. At the end of the job, set the write-address mode switch off.

To use the deck when the system is to reside on 1301:

1. Set the write-address mode switch on.
2. Set the write-disk switch on.
3. Set the I/O check stop switch on.
4. Press CHECK RESET and START RESET.
5. Place the Write File-Protected Addresses section in the card reader.
6. Load the program.
   a. 1402 Card Reader: Press LOAD.
   b. 1442 Card Reader: Press START on the reader, and PROGRAM LOAD on the console.
7. When the system attempts to read the last card,
   a. 1402 Card Reader: Press START.
   b. 1442 Card Reader: Press START on the reader.
8. At the end of the job, set the write-address mode switch off.

The following halts can occur when writing file-protected addresses.

Halt Number (A-Address Register)	*Meaning*
020	Last card condition was sensed before the control card. The control card containing the initial and terminal addresses of the area to be file-protected must be the last card of the deck. When the system is restarted by pressing START, a read operation is performed.
021	An invalid disk type is specified in the control card. 1301 or 1311 are the only valid entries for columns 17-20 of the control card. When the system is restarted by pressing START, a read operation is performed.
022	An invalid disk unit is specified in the control card. The only valid entry for column 22 of the control card is 0. When the system is restarted by pressing START, a read operation is performed.
023	An invalid start address (columns 44-49) is specified in the control card. The start address must be 002500. When the system is restarted by pressing START, a read operation is performed.
024	An invalid end address (columns 54-59) is specified in the control card. The end address must be 004800. When the system is restarted by pressing START, a read operation is performed.
025	Disk unit 0 is not ready. When the system is restarted by pressing START, the disk I/O operation is retried.
026	The area specified in the control card is already file-protected (all or in part). If the system is restarted by pressing START, the entire specified area will be file-protected *and* cleared.
027	The area specified in the control card has neither the "normal" disk addresses (000000-?) nor file-protected addresses. This is a hard halt.
028	Parity check or wrong-length record error occurred on the disk unit while writing addresses. When the system is restarted by pressing START, the disk I/O operation is retried.
029	Parity check or wrong-length record error occurred on the disk unit while determining the existing addressing scheme. This is a hard halt.
030	End of the job.

System Control Card Build

The last card in the section labeled CARD BUILD is a control card that is partially prepunched. It is by the use of this control card that disk residence is determined.

The user must indicate in the control card whether the System is to reside on a 1301 or 1311 disk unit. The assumed disk unit number is 0.

The control card is punched as follows:

Columns	*Contents*
6-11	SYSTEM (prepunched)
16-20	BUILD (prepunched)
21-24	1301 or 1311

After columns 21-24 have been punched by the user, the card must be replaced as the last card of the CARD BUILD deck.

The System Control Card Build consists of the card sections labeled CARD BUILD and SYSTEM CONTROL.

To use the System Control Card Build when the system is to reside on 1311:

1. Ready the pack on disk drive 0.
2. Set the write-address mode switch off.
3. Set the write-disk switch on.
4. Set the I/O check stop switch off.
5. Press CHECK RESET and START RESET.
6. Place the System Control Card Build section in the card reader.
7. Load the program.
   a. 1402 Card Reader: Press LOAD.
   b. 1442 Card Reader: Press START on the reader, and PROGRAM LOAD on the console.

8. When the system attempts to read the last card,
   a. 1402 Card Reader: Press START.
   b. 1442 Card Reader: Press START on the reader.

To use the System Control Card Build when the System is to reside on 1301:

1. Set the write-disk switch on.
2. Set write-address mode switch off.
3. Set the I/O check stop switch off.
4. Press CHECK RESET and START RESET.
5. Place the System Control Card Build section in the card reader.
6. Load the program.
   a. 1402 Card Reader: Press LOAD.
   b. 1442 Card Reader: Press START on the reader, and PROGRAM LOAD on the console.
7. When the system attempts to read the last card,
   a. 1402 Card Reader: Press START.
   b. 1442 Card Reader: Press START on the reader.

The following halts can occur while using the System Control Card Build deck.

Halt Number (A-Address Register)	*Meaning*
008	Card-read error: To retry the operation, *For the 1442:* Non-process run-out the cards. Place the two non-processed cards in the read hopper. Press START on the reader and START on the console. *For the 1402:* Non-process run-out the cards. Remove the last three cards in the stacker and place them in the hopper. Press START.
050	The SYSTEM BUILD control card is missing from the deck or the user entry is incorrectly punched.
051	End of job.
549	Disk unit 0 is not ready. When the system is restarted by pressing START, the disk I/O operation is retried.
554	A disk-write error occurred ten times. When the system is restarted by pressing START, the disk I/O operation is retried.

Autocoder Update

To build the Autocoder Assembler, the Autocoder Update Section, made up of the sets of cards labeled AUTOCODER PROCESSOR, AUTOCODER PREPROCESSOR, and AUTOCODER MACROS are used. (This section, labeled AUTOCODER PROCESSOR, contains the Autocoder, Output, and Execution processors.)

Note: If it is necessary to rebuild the Preprocessor, the user can avoid destroying the LIBRARY file that is within the limits assumed by the Autocoder System. To do this, place a *dummy* LIBRARY ASGN card ahead of the AUTOCODER RUN card which is the first card in the AUTOCODER PREPROCESSOR section. This *dummy* ASGN card should specify a disk area whose contents need not be saved. For example, the area that is allotted to WORK1 could be specified.

Input for this building process is as follows:

1. The 1402 or 1442 Card Boot followed by
2. The SYSTEM ASGN card, which must be punched by the user, followed by
3. The Autocoder Update Section followed by
4. The HALT card, which must be punched by the user.

To build the System when it is to reside on 1311:

1. Ready the pack on disk drive 0.
2. Set the I/O check-stop switch off.
3. Set the check-stop switch and disk-write switch on.
4. Set the mode switch to RUN.
5. Press CHECK RESET and START RESET.
6. Load the program.
   a. 1402 Card Reader: Press LOAD.
   b. 1442 Card Reader: Press START on the reader, and PROGRAM LOAD on the console.
7. When the system attempts to read the last card,
   a. 1402 Card Reader: Press START.
   b. 1442 Card Reader: Press START on the reader.

To build the system when it is to reside on 1301:

1. Set the I/O check-stop switch off.
2. Set the check-stop switch and disk-write switch on.
3. Set the mode switch to RUN.
4. Press CHECK RESET and START RESET.
5. Load the program.
   a. 1402 Card Reader: Press LOAD.
   b. 1442 Card Reader: Press START on the reader, and PROGRAM LOAD on the console.
6. When the system attempts to read the last card,
   a. 1402 Card Reader: Press START.
   b. 1442 Card Reader: Press START on the reader.

The halts that can occur when using the Autocoder Update deck are shown in Figure 26.

Sample Program

The Sample Program, which is used to test the effectiveness of the system built by the user, calculates and lists a table of salaries. A listing of the Sample Program is shown in Appendix III.

The first card in the sample program is a partially prepunched control card used for assigning the CORELOAD file.

The user must indicate in the control card whether the system resides on a 1301 or 1311 disk unit. The control card is punched as follows:

Columns	*Contents*
6-13	CORELOAD (prepunched)
16-19	ASGN (prepunched)
21-24	1301 or 1311
26-57	UNIT 0, START 012300, END 012399 (prepunched)

To prepare and run the Sample Program, see *Preparing a Stack* and *Running a Stack*. Figure 31 shows the sample program job deck.

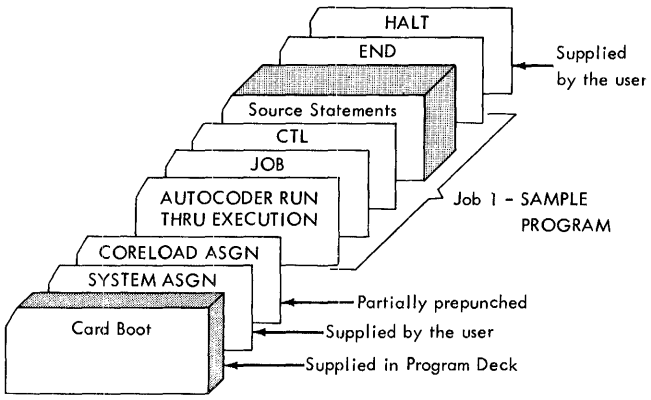

Figure 31. Sample Program

Updating an Autocoder System

The Autocoder System is updated by the use of prepunched card decks supplied by IBM. All necessary control cards and data cards are included in the deck.

An update job is performed as described in *Preparing a Stack* and *Running a Stack*.

Copying an Autocoder System

The Autocoder System can be copied by the IBM Copy Disk utility program using the information provided in this section if: (1) the Autocoder System resides on 1311 disk storage; (2) the LIBRARY and WORK file assignments are the same as was initially assumed by the System Control Program (see Figure 32); and (3) the system pack to be copied does not contain the COBOL or Fortran System in addition to the Autocoder System.

The procedure for copying the Autocoder System is as follows:

1. Mount the Autocoder System disk pack on (1311) unit 0. Mount a disk pack on (1311) unit 1. (The pack on unit 1 will contain the copy of the Autocoder System.)
2. Using the Clear Disk utility program, clear the following areas of the pack on unit 1. (The mode of operation, the *drive* number being used, and the drive to which addresses are referenced are also given to aid in punching the area-control cards needed by the Clear Disk program).

Mode	*Lower Limit*	*Upper Limit*	*Drive Used*	*Drive Referenced*
M	000000	000199	2	0
L	000200	000259	2	0
M	000260	000299	2	0
L	000300	004799	2	0
M	004800	019979	2	0

3. Using the Copy Disk Utility program, copy the following areas from the pack on unit 0 to the pack on unit 1. (The mode of operation, the number of the drive from which the Autocoder System is written, the number of the drive to which the System is copied, and the drive to which the addresses are referenced are also given to aid in punching the area-control cards needed by the Copy Disk program.)

Mode	*Lower Limit*	*Upper Limit*	*Drive From*	*Drive To*	*Drive Referenced*
L	000200	000259	0	2	0
M	000260	000299	0	2	0
L	000300	002499	0	2	0
L	262500	264799	0	2	0
M	012900	019979	0	2	0

The program specifiications and the operating procedures for the Clear Disk and Copy Disk utility programs are given in the following publications:

1. *Disk Utility Programs Specifications,* IBM *1401, 1440, 1460 (with 1301 and 1311),* Form C24-1484.
2. *Disk Utility Programs Operating Procedures,* IBM *1401 and 1460 (with 1301 and 1311),* Form C24-3105.
3. *Disk Utility Programs Operating Procedures,* IBM *1440-1311,* IBM *1440-1301,* Form C24-3121.

Appendix I

This section contains a summary of the formats of all control cards that are required for System operations. Each control card is punched in the Autocoder format (the label field is in columns 6-15, the operation field is in columns 16-20, and the operand field is in columns 21-72).

The user is again reminded that in columns 21-72, blanks must appear as indicated in the individual formats.

Figure 32 shows the formats of ASGN cards and the assumed assignments for the logical files. Figure 33 shows the valid device entries for the ASGN cards.

Figure 34 shows the formats of the following control cards:

Halt (HALT) card
Init (INIT) card
Librarian-control cards
Note (NOTE) card
Output option (OPTN) cards
Pause (PAUSE) card
Run (RUN) cards.

Note: Update cards are prepunched and included in the card decks supplied by IBM for updating the user's System.

ASGN Card Format			Assumed Assignment	Remarks
Label Field (Columns 6-15)	Operation Field (Columns 16-20)	Operand Field (Columns 21-72)		
SYSTEM	ASGN	{1311 UNIT n / 1301 UNIT 0}	1311 unit -- user-assigned 1301 unit -- must be assigned to UNIT 0	The SYSTEM ASGN card is the only required ASGN card. It must follow the Card Boot in a stack of jobs. Any other SYSTEM ASGN cards in the stack are invalid. If the user desires that the Autocoder System use less than the number of core-storage positions available in the processor machine, punch a comma in column 32 and 4K, 8K, 12K, or 16K beginning in column 34.
CONTROL	ASGN	{READER n / CONSOLE PRINTER}	READER 1	
MESSAGE	ASGN	{PRINTER n / CONSOLE PRINTER}	PRINTER 2	When the MESSAGE file is assigned to the CONSOLE PRINTER, carriage control characters used with the 1403 or 1443 printer may appear in the message.
LIST	ASGN	{PRINTER n / 1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn / OMIT}	PRINTER 2	If the LIST file is assigned to PRINTER 1 (1403), the Output processor develops a 100-character program listing.
INPUT	ASGN	{READER n / 1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn}	READER 1	
OUTPUT	ASGN	{PUNCH n / 1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn / OMIT}	PUNCH 4 (1401 and 1460) PUNCH 1 (1440)	
LIBRARY	ASGN	{1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn}	{1301 / 1311} UNIT 0, START 012900, END 019980	1311 is assumed if the SYSTEM file is assigned to 1311; 1301 is assumed if the SYSTEM file is assigned to 1301.
WORK1	ASGN	{1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn}	{1311 / 1301} UNIT 0, START 004800, END 011200	
WORK2	ASGN	{1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn}	{1311 / 1301} UNIT 0, START 011200, END 012400	
WORK3	ASGN	{1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn}	{1311 / 1301} UNIT 0, START 012400, END 012900	
CORELOAD	ASGN	{1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn / OMIT}	OMIT	

NOTE: If the user's system contains Autocoder and COBOL, the WORK1 assumed assignment is changed from START 004800, END 011200 to START 007200, END 010400. The assumed assignments for WORK2 and WORK3 remain the same.

Figure 32. ASGN Card Formats and Assumed Assignments

Device Entry and Values of n and nnnnnn	Remarks
{1311 / 1301} UNIT n, START nnnnnn, END nnnnnn n is the number of the disk unit, and can be 0, 1, 2, 3, or 4; nnnnnn is a disk address.	The END address is the address of the next available sector. The values of nnnnnn must adhere to the following rules: 1. WORK1 file. If the disk unit is 1311, the START address must be a multiple of 200. If the disk unit is 1301, the START address must be a multiple of 800. The END address (1311 and 1301) must be a multiple of 40. 2. WORK2 and WORK3 files. The START address (1311 and 1301) of each file must be a multiple of 100. (A START address that is a multiple of 200 results in the fastest assembly.) The END address (1311 and 1301) of each file must be a multiple of 10. 3. LIBRARY file. The START and END addresses (1311 and 1301) must be multiples of 20. 4. OUTPUT file. It is not necessary to specify that this file start or end at any particular multiple. However, Autocoder will only use the file if it begins at a multiple of 40. In each of the first three cases, if the rules are violated, the system automatically narrows in the disk area to an area that does adhere to the rules. Incorrect addresses are not automatically corrected for the OUTPUT file.
READER n For 1402, n can be 0, 1, or 2. For 1442, n can be 1 or 2.	 For 1402, n represents the pocket into which the cards are stacked. For 1442 and 1444, n represents the number of the unit.
PUNCH n For 1402, n can be 0, 4, or 8. For 1442, n can be 1 or 2 For 1444, n must be 3.	
PRINTER n n can be 1 or 2	n represents the number of print positions available on the 1403 or 1443. For 1403, a 1 indicates 100 positions and a 2 indicates 132 positions. For 1443, a 1 indicates 120 positions and a 2 indicates 144* positions. * Only 132 print positions are used by the Autocoder System.
CONSOLE PRINTER	The console printer for the control file must be an IBM 1447 without a buffer feature or an IBM 1407. An IBM 1447 with a buffer feature can be used for the message file, although the buffer feature is not used.
OMIT	Select this option when the file is not to be used by the Autocoder System. LIST, OUTPUT, and CORELOAD are the only files that can be omitted.

Figure 33. Valid Device Entries

Name of Card	Label Field (Columns 6-15)	Operation Field (Columns 16-20)	Operand Field (Columns 21-72)
Halt		HALT	Any message and/or identification
Init		INIT	Any message and/or identification
Library Control	INITIALIZE	OPTN	Blank
	LIBRARY	OPTN	1 (for direct seek), or blank
	COMPAT	OPTN	1 (for direct seek), or blank
	LISTING	OPTN	ALL
	LISTING	OPTN	HEADER
	LISTING	OPTN	Blank
	LISTING	OPTN	PUNCH
	LISTING	OPTN	PUNCH1444
		Macro name	(Used if columns 21-72 of the LISTING OPTN card are blank)
	Macro name	INSER	n (insertion) m, n (substitution) Blank (insertion of entire routine)
	Macro name	DELET	n (deletion of one statement) m, n (deletion of two or more statements) Blank (deletion of entire routine)
		END	
Note		NOTE	Any message an/or instruction
Output Option	LIST	OPTN	nn (nn is the number of lines per page)
	PUNCH	OPTN	x (x is S for self-loading format) and blank for condensed-loader format)
	RESEQ	OPTN	
	CORELOAD	OPTN	
Pause		PAUSE	Any message and/or instruction
Run	AUTOCODER	RUN	
	AUTOCODER	RUN	THRU OUTPUT
	AUTOCODER	RUN	THRU EXECUTION
	OUTPUT	RUN	
	OUTPUT	RUN	THRU EXECUTION
	EXECUTION	RUN	Note: Information in addition to any of the above entries should not be used.

Figure 34. Control Card Formats

The name, identification, and function of each phase in the Autocoder System are given in the following sections.

System Control Program

This section describes the phases that make up the System Control Program.

Name	*ID*	*Function*
Card Build	50X41 (1442) 50X01 (1402)	Builds System Control on 1311 or 1301.
Card Boot	50SZ1 (1442) 50PZ1 (1402)	Read the SYSTEM ASGN card and reads in the System Boot from the specified disk unit.
System Boot	50S01	1. Determines machine size. 2. Initializes switches according to the type of reader, punch, and printer (serial or parallel). 3. Reads in the I/O package. 4. Calls the determiner.
File-Hardware Table	50S11	Contains the assumed assignments for the logical files.
Input/Output Package	50S21	1. Reads or writes disk in the move or load mode. The mode depends on the processor operation. 2. Determines whether the user has exceeded specified file limits. 3. Branches to the processor phase, or branches to the end-of-file routine if the end-of-file has been sensed.
Super 0	50S31	Reads in the specified phase from disk storage and branches to the specified phase.
Super 1	50S41	
Super 2	50S51	
Super 3	50S61	
Super 4	50S71	
Super 5	50S81	
Super 6	50S91	
Open 1	50SA1	Initializes the specified area with a twenty-character control word. This control word is obtained from the temporary file-hardware table.
Open 2	50SB1	
Determiner	50SC1	Reads the CONTROL file until a control card (HALT, PAUSE, NOTE, UPDAT, RUN, or ASGN) is sensed. When a control card is sensed, the determiner causes a halt or pauses, prints out a note, calls the update determiner, calls the selector, or calls the configurator, depending upon the type of card.
Phase Index Table	50SD1	Contains the locations of the phases in the System.
Configurator	50SE1	Updates the temporary file-hardware table as specified by the ASGN card(s.)
Selector	50SF1	Initializes the files used by the processor being called, and calls the first phase of that processor.
Update Determiner	50SG1	Determines the type of update operation being performed, and calls in that particular updater.
Update Insert	50SH1	Places a new phase on the SYSTEM file in any available location.
Update Header	50SI1	Updates the header of a phase that is in the SYSTEM file, as specified by a header card.
Update Delete	50SJ1	Deletes a phase from the SYSTEM file.
Update Patch	50SK1	Patches a part of a phase on the SYSTEM file.
Dump 1	50SL1	Prints storage on the LIST file.
Dump 2	50SM1	
File Print 1	50SN1	Prints all WORK files on the LIST file.
File Print 2	50SO1	
File Print 3	50SP1	

Note: The Dump and File Print phases are used only if the SYSTEM file contains the COBOL compiler.

Preprocessor and Autocoder Processor

This section describes the phases that make up the Preprocessor and the Autocoder processor.

In the discussion of the Autocoder processor phases (initialize, mnemonic, conditioner, diagnostic, literal, label-entry, symbol-lookup, and pre-output), *text* refers to a series of 100-character records, each of which contains an 80-character source portion and a 20-character object portion. Source program and generated statements are written in the *source portion* by the Preprocessor; machine language instructions are built in the *object portion* by the Autocoder processor.

Name	*ID*	*Function*
Initialize	AU1AA	1. Opens all files used by the Preprocessor, and saves the addresses of disk files. 2. Saves all constants that have to be returned to System control. 3. Calls AU1BA if it is needed. 4. Check the OUTPUT and LIBRARY file assignments. 5. Calculates the number of blocks available in the LIBRARY file. 6. Calls option-control (RUNCL).
Build	AU1BA	1. Loads the phase table, dummy library, option-control phase (RUNCL), monitor phase (MNTOR), and update phase (UPDAT) into the SYSTEM file. The phase table contains the entry for the update phase and a trailor flag which indicates the end of the table. The dummy library (DUMLB) creates the LIBRARY file. It builds the library table and a record that specifies the end-of-library name (99999). The end-of-library name and its disk address are written in the library table. 2. Calls in RUNCL when an Autocoder option card is encountered.
Linkage	DI000	1. Reads in the System Control Program. 2. Inserts saved constants. 3. Puts a $ in location 3999 if a macro generation has been completed. 4. Calls the selector phase of the System Control Program.
Pre-phase	DIOCS	Sets up parameters for DIOCS entries.
Pre-phase	DTF	Sets up parameters for DTF entries.
Pre-phase	GT-PT	Sets up parameters for GET and PUT phases.
Pre-phase	GENPP	Sets temporary switches for DIOCS, DTF, GT-PT, FILEORG, and SORT parameters.
Option Control	RUNCL	1. Clears storage (except the storage positions containing RUNCL). 2. Reads the first record (card or card image) in the INPUT file. 3. Moves the phase name (columns 6-10) to the permanent area if the record contains an option, otherwise, calls AUTOC. 4. Permanent monitor calls MNTOR (between phase monitor).
Between-phase Monitor	MNTOR	1. Reads in the phase-header table and selects the entry for the phase name written in the permanent monitor area. 2. Initializes the permanent monitor to read in the required phase giving this information: a. The terminating group-mark word-mark b. The starting disk sector c. The number of sectors d. The clear-storage address e. The read-into-core address f. The execute-to-phase address g. The split-phase suffix code. 3. If requested, clears storage from the address specified in the phase header to 1000. 4. If the phase header has a split-phase suffix, initializes the permanent monitor to return to the between-phase monitor after reading the first part of the phase. 5. Upon return from the permanent monitor, moves suffix to the phase-name request location in the permanent monitor and transfers back to step 1 for the next part of the phase.
Update	SYSTE	1. Replaces entries in the phase-header table with new (modified) entries. 2. Adds new entries to the phase-header table. 3. Obtains the corresponding phase-header table entry when an UPDAT card is encountered. Also clears the disk sectors used by the phase to load mode blank. For the new condensed deck, performs the same procedure as described for UPDAT PATCH in item 4. 4. For each condensed card following an UPDAT PATCH, computes the sector(s) to be patched and the starting location within the sector(s). Also, it reads in the sector(s), performs the patch by simulating an object-deck loader, writes the sectors back on the SYSTEM file, and proceeds to the next condensed card. 5. Calls in DI000 after the update operation has been completed.

Name	*ID*	*Function*
Library Build	INITI	1. Writes a record that contains the end-of-library name (99999). 2. Writes the end-of-library name and its disk address in the library table.
Library Change	LIBRA	1. Reads in the library table. 2. Obtains, from the table, the beginning and ending disk-sector addresses of the existing library. 3. Sets up direct-seek test flag for LIBR2.
	LIBR2	4. Recopies the entire existing library. During this process, all insertions and deletions are made. All inserted statements are rebuilt to conform to the disk Autocoder library format. 5. Writes the routine name and the beginning disk-sector address in the library table. 6. Writes the routine on the LIBRARY file. 7. Repeats steps 5 and 6 until all routines have been written. The new library is written starting one sector beyond the end of the old library. The library area is effectively a closed loop. When the upper area is reached, the remaining routines are written starting at the lower limit.
	LIBR4	8. Writes the new library table in the LIBRARY file. 9. Calls in DI000 (linkage) after the library change operation has been completed.
Library Listing	LISTI	1. Reads in the library table. 2. Prints all of the names in the library table if the LISTING OPTN card specifies HEADERS. 3. Prints the entire library if the LISTING OPTN card specifies ALL. 4. Prints the specified routines if routine-name cards follow the LISTING OPTN card. The starting address of each specified routine is obtained from the library table. 5. Punches the entire library into cards with each routine preceded by an INSER card if the LISTING OPTN card specifies PUNCH or PUNCH1444. 6. Calls in DI000 after the listing or punching operation has been completed.
Constant Storage	AUTOC	1. Prepares the WORK1 file to use 8K (larger blocks) if more than 4K storage is available. 2. Initializes all areas and output routines needed during the macro pass. 3. Reads the JOB, CTL, and comments cards and writes them on the WORK1 file. 4. Analyzes the CTL card. Sets up permanent switches for the macro generator (MAGEN). These indicate the size of the object machine, the presence of the modify-address, advanced-programming or indexing-and-store-address-register, and multiply-divide features. 5. Sets up a sumbox for returning to the program loader during object-program execution. 6. Sets up the macro pass modify-add procedure.
Main Line	MNLIN	1. Reads source statements. 2. Writes all source statements, except DTF entries, on the WORK1 file. 3. Calls SPSCV (conversion phase) if an ENT statement is encountered. 4. Calls in the DTF phase upon encountering a DTF entry. 5. Places source INCLD statements in the INCLD file for processing at LTORG, EX, or END time. (The INCLD file is located on the WORK3 file.) 6. Processes the CHAIN macro developing as many object-program statements containing the specified operation code as are stated in the operand field of the CHAIN macro. 7. Sets up the source macro instructions for MAGEN (macro generator). a. Turns off all temporary switches and adds 1 to the □0J counter. b. Turns on the particular temporary switches for which there is a parameter. c. Relocates the parameter for use by the macro generator. d. Counts the parameters. e. Writes out source macro statements as comments. f. Generates and writes out LABEL EQU *+1 if the source macro statement has a label. g. Extracts disk address of library routine from library table for MAGEN. h. Calls the macro generator and transfers control to it. 8. Handles MA instructions as macros if the Modify Address feature is not available on the object machine. 9. For an EX statement, the MNLIN checks a special counter to determine if there is a closed library routine in the INCLD file. If the

Name	*ID*	*Function*
		counter is zero, it writes out the statement and proceeds to the next card. If the counter is not zero. it calls in MCIMC (macro-in-macro) and transfers control to it. 10. For a LTORG statement in any form other than LTORG *, the MNLIN replaces the LTORG mnemonic with ORG and writes it on WORK1. From this point, the procedure described in step 9 is taken. Upon return, a LTORG * statement is generated and inserted in WORK1. 11. For an END statement the MNLIN follows the same procedure described in step 9. 12. Calls AUA/C (first assembler phase) after END has been written on WORK1.
Conversion	SPSCV	1. Reads source statements written in 1401 SPS or 1440 Basic Autocoder format. 2. Converts the statements to disk Autocoder format. 3. Writes the converted statements on WORK1. 4. Calls MNLIN (main line) when an ENT AUTOCODER statement is encountered.
Macro Generator	MAGEN	1. Extracts the next sequential library statement (see step 7g under MNLIN). 2. If this statement is a macro header, MAGEN assumes an end-of-macro condition and calls in MNLIN (main line). 3. If a BOOL skip condition is in effect, MAGEN tests for the BOOL label character. If the required BOOL label character is not in this statement, the program returns to item 1; otherwise, it turns off the BOOL skip switch and proceeds. 4. If the statement specifies MEND, MAGEN calls MNLIN (mainline). 5. Scans the entire record for □ and # symbols substituting parameters and sumboxes in model statements. If □0J type symbols are encountered, the program inserts the contents of the macro counter beside the symbol (for example, □0J023). The values of temporary or permanent switches are substituted in BOOL statements. 6. If the statement is an INCLD or a macro-in-macro type, the program places it on the INCLD file for later processing and writes it as a comment on WORK1, MAGEN then returns to step 1. 7. If the statement is not a MATH or BOOL, MAGEN writes it on WORK1 and returns to step 1. 8. Solves MATH and BOOL statements. 9. Sets any switches specified to show results of processing MATH or BOOL statements. 10. Places the result developed for a MATH statement in the specified sumbox (if any) and goes to step 7. 11. If the statement is a BOOL and has a true result, MAGEN returns to step 1. If the result is false, MAGEN sets up a BOOL skip if a BOOL label character is specified.
Macro-in-Macro	MCIMC	1. Reads the INCLD file. 2. Handles macros as described under MNLIN (main line). 3. Writes all non-macro statements (not named in the library table), except INCLD, on WORK1. 4. Calls in INCLD (include) after all statements have been processed.
Include	INCLD	1. Reads the INCLD file. 2. Bypasses all statements except INCLD statements. 3. Write INCLD statements on WORK1 as comments. 4. Writes the library routine specified by each INCLD statement on WORK1. If a duplicate INCLD statement is encountered, its associated library routine is not extracted because library routines for INCLD statements are included in the object program only once per program overlay. 5. Eliminates the contents of the file and resets the address of the file after all the statements in the INCLD file have been processed. Calls MNLIN (main line).
Initialize	AUA/C	1. Sets up I/O areas according to machine size. 2. Initializes I/O routines.
Mnemonic	AU3AA AU3BA AU3CA AU3SA	1. Reads source statements (text) from WORK1. 2. Analyzes the CTL card. Prints the CTL card image and diagnostic messages if any characters are invalid. 3. Looks up mnemonic operation codes and inserts the machine-language operation codes, d-modifiers, control

Name	*ID*	*Function*
		fields, and operand-type codes in the object portion. 4. Inserts flags to indicate the type of statement (assembler-control, comment, instruction). 5. Inserts a diagnostic flag symbol in the object portion if an operation code is invalid. 6. Writes the text on WORK1.
Conditioner	AU4AA AU4BA AU4CA AU4DA AU4EA	1. Reads the text from WORK1. 2. Extracts operands from the source portion and inserts tags in the object portion. 3. Interprets indexing and/or adjustments and inserts tags in the object portion. 4. Extracts symbolic indexing and develops symbolic indexing table. 5. Inserts a diagnostic flag in the object portion if any errors are detected. 6. Writes the text on WORK1.
Diagnostic	AU4WA AU4XA	1. Reads text from WORK1. 2. Analyzes operand errors according to operand-type codes. 3. Checks d-modifiers in the source portion against a valid d-modifier list. 4. Checks the label field. 5. Writes diagnostic messages and incorrect source statements on the MESSAGE file.
Literal	AU5AA AU5JA AU5KA	1. Reads text from WORK1. 2. Extracts literals, checks for duplication, and creates literal table. 3. Purges literal table and writes, on WORK1 (literal file), the processor-declared literal fields for each program overlay. 4. Inserts the displacement integer in the object portion for each entered literal. 5. Writes text on WORK1.
Symbol Extraction	AU5MA AU5NA	1. Extracts symbols (labels and area-defining literals) addresses and sequence numbers. 2. Builds the cross reference table on WORK2.
Label Entry	AU7DA AU7EA AU7FA AU7GA	1. Reads text and literal file from WORK1. 2. Extracts labels from the source and enters them in the label table. 3. Enters all literal origin addresses into the literal-base table. 4. Enters EQU labels into the label table if previously defined, or enters symbolic EQU labels into the EQU table if not previously defined. 5. Assigns addresses and allocates areas for instructions, defined areas, and constants.
Sort and Merge	AU7HA AU7IA AU7JA AU7KA AU7MA AU7NA AU7OA	After the symbol table has been built on WORK2 or the label table on WORK3: a. Arranges symbols or labels in alphabetical order within segments of the label on symbol table. b. Merges table segments to alphabetize the entire table.
Table Analyzers	AU7UA AU7XA	1. Converts symbolic EQU entries to actual addresses. 2. Detects and flags error conditions in the table. 3. Creates reference table for use by the symbolic lookup.
Symbol Lookup	AU7SA AU7TA	1. Reads text and literal file. 2. Looks up symbols in the label table and inserts actual addresses in the object portion. 3. Adds the literal bases found in the table to the displacement developed during the literal phase. 4. Inserts a diagnostic flag symbol in the object portion if any symbols are multiply defined or undefined. 5. Writes text and literal file on WORK1.
Pre-output	AU8AA	1. Reads text and literal file from WORK1. 2. Extracts adjustments and/or indexing from the source portion and adjusts the actual addresses. 3. Conditions the object portion of text for processing by the OUTPUT processor. 4. Inserts a diagnostic flag symbol if any errors are detected. 5. Writes text on the OUTPUT file. WORK1 becomes OUTPUT if the THRU option is specified in the AUTOCODER RUN card.
Label Table Listing	AU7XA	Writes the label table listing on the LIST file.
Cross Reference Listing	AU9AA AU9BA	Writes the cross reference listing on the LIST file.

Output Processor

This section describes the phases that make up the Output processor.

ID	*Function*
OP1OP	1. Sets up I/O areas according to machine size. 2. Initializes I/O routines. 3. Opens required files.
OP2OP	Reads the INPUT file and transfers the program listing to the LIST file.

ID	*Function*
OP2OQ	Writes the end-of-listing message.
OP3OP	1. Transfers the clear-storage and bootstrap routines for the 1442 to the OUTPUT file. 2. Transfers the 1442 loader to the OUTPUT file.
OP4OP	1. Reads the INPUT file.
OP4OQ	2. Processes the source.
OP4OR OP4OS OP4OT	3. Transfers the object program in condensed loader.
OP5OP	1. Reads the INPUT file. 2. Resequences source-program statements. 3. Transfers the resequenced statements to the OUTPUT file.
OP6OP	Builds 90-character sectors in the CORELOAD file.
OP6OQ	Builds operating sectors in the CORELOAD file.
OP6OR OP6OS OP6OT OP6OU	1. Reads the INPUT file. 2. Loads the object portion of the source into the CORELOAD file.
OP7OP	1. Reads the INPUT file.
OP7OQ	2. Processes the source.
OP7OR OP7OS OP7OT	3. Transfers the self-loading object program to the OUTPUT file.
OP8OP	Transfers the clear-storage and bootstrap routines for the 1402 to the OUTPUT file. Transfers the 1402 loader to the OUTPUT file.
OP9OP	Reads the output option cards. Selects the phases to be used.

Execution Processor

This section describes the phases that make up the Execution processor.

ID	*Function*
EXEXC	Sets up I/O areas. Initializes I/O routines. Checks header record in the INPUT file. Clears storage. Initializes disk loader and bootback routines.
EX4EX	Disk loader and bootback for 4K.
EX8EX	Disk loader and bootback for 8K.
EX2EX	Disk loader and bootback for 12K.
EX6EX	Disk loader and bootback for 16K.

This section contains a listing of the sample program (Figure 35) that is the last part of the Autocoder System Program Deck. This program is designed to calculate the hourly, weekly, and annual salaries associated with a given monthly salary. The monthly salary starts at $400 and is increased by $25 until it equals $900.

```
                    CORELOAD  ASGN 1311 UNIT 0, START 012300, END 012399
                    AUTOCODER RUN  THRU EXECUTION
```

CROSS REFERENCE LISTING

ADDRS	LABEL	TAG	SEQUENCE NUMBERS OF INSTRUCTIONS WHERE SYMBOL APPEARS									
00608	¤0J001		0029	0033	0036							
00633	¤0K001		0034	0037								
00747	¤0K002		0058	0063								
00970	¤0K003		0098	0103								
00641	¤0L001		0036	0028								
00781	¤0L002		0064	0068	0072							
01004	¤0L003		0104	0108	0112							
00667	¤0M001		0041	0030	0030	0034	0044					
00805	¤0M002		0069	0065								
01028	¤0M003		0109	0105								
00660	¤0N001		0040	0027	0029	0033	0036					
00658	¤0P001		0039	0034								
00839	¤0P002		0074	0058	0069	0070	0070					
01062	¤0P003		0114	0098	0109	0110	0110					
00667	¤0Q001		0042	0024								
00848	¤0Q002		0075	0051	0052	0059	0060	0062	0064	0066	0071	0071
01071	¤0Q003		0115	0091	0092	0099	0100	0102	0104	0106	0111	0111
00668	¤0R001		0043	0029								
00874	¤0R002		0079	0058								
01097	¤0R003		0119	0098								
00883	¤1J002		0081	0059	0060							
01106	¤1J003		0121	0099	0100							
00856	¤1K002		0076	0057	0083	0084						
01079	¤1K003		0116	0097	0123	0124						
00865	¤1L002		0077	0053	0054	0055	0056	0062	0064	0066		
01088	¤1L003		0117	0093	0094	0095	0096	0102	0104	0106		
00884	¤1N002		0082	0069								
01107	¤1N003		0122	0109								
01411	ACCUM		0183	0044	0046	0049	0054	0084	0086	0089	0094	0124	0126
			0128										
01404	AREA		0182	0018	0019	0020	0021	0045	0046	0047	0048	0085	0086
			0087	0088	0125	0126	0127	0129	0144	0145	0146	0147	0148
			0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
00500	BEG		0004	0191								
01447	FIFTH		0189	0051								
01449	FORTY		0190	0091								
01381	HOUR		0179	0014								
01390	MASK		0180	0018	0045	0085	0125	0144	0148	0152	0156	
01363	MONTH		0176	0011								
01395	MTH		0181	0019	0022	0026	0132	0133				
00553	START		0018	0134								
01419	TOT1		0184	0022	0145							
01427	TOT2		0185	0049	0149							
01435	TOT3		0186	0089	0153							
01443	TOT4		0187	0128	0157							
01445	TWLV		0188	0025								
01375	WEEK		0178	0013								
01369	YEAR		0177	0012								

Figure 35. Sample Program (Part 1 of 6)

```
SPL01                    SALARY TABLE COMPUTATIONS                    PAGE    1

SEQ  PGLIN LABEL   OPCD  OPERAND                          SFX CT  LOCN  INSTRCTN A-ADD B-ADD .FLAGS

0001 0001  SPL01   JOB     SALARY TABLE COMPUTATIONS                                          .
0002 0002          CTL   1                                                                  .
0003 0005          ORG   500                                        500                     .
0004 0006  BEG     CS    332                                 4    500  /332       332      .
0005 0007          CS                                        1    504  /                    .
0006 0008          MLC   @SALARY TABLE@,241                  7    505  MU61241    1461  241.
0007 0009          W                                         1    512  2                    .
0008 0010          CS    299                                 4    513  /299       299      .
0009 0011          W                                         1    517  2                    .
0010 0012          W                                         1    518  2                    .
0011 0013          MLC   MONTH,216                           7    519  MT63216    1363  216.
0012 0014          MLC   YEAR,232                            7    526  MT69232    1369  232.
0013 0015          MLC   WEEK,247                           7    533  MT75247    1375  247.
0014 0016          MLC   HOUR,263                           7    540  MT81263    1381  263.
0015 0017          W                                         1    547  2                    .
0016 0018          CS    299                                 4    548  /299       299      .
0017 0019          W                                         1    552  2                    .
0018 0020  START   MLCWA MASK,AREA                          7    553  LT90U04    1390  1404.
0019 0021          MCE   MTH,AREA                            7    560  ET95U04    1395  1404.
0020 0022          SW    AREA-8                              4    567  ,T96       1396       .
0021 0023          MLC   AREA,216                            7    571  MU04216    1404  216.
0022 0024          A     MTH,TOT1                            7    578  AT95U19    1395  1419.
0023 0025            MLTPYTWLV,2,0,MTH,5,2,ACCUM,7,2                   **MACRO**            .
0024 *             S     ¤0Q001                              4    585  S667       667      .
0025 *             MLCWA TWLV                                4    589  LU45       1445     .
0026 *             MLCWA MTH                                 4    593  LT95       1395     .
0027 *             S     &0,¤0N001                           7    597  SU62660    1462  660.
0028 *             B     ¤0L001                              4    604  B641       641      .
0029 *     ¤0J001  BW    ¤0R001,¤0N001                       8    608  V6686601   668   660.
0030 *             MLC   ¤0M001-1,¤0M001                     7    616  M666667    666   667.
0031 *             MLNS                                      1    623  D                    .
0032 *             MLCWA                                     1    624  L                    .
0033 *             BCE   ¤0J001,¤0N001,0                     8    625  B6086600   608   660.
0034 *     ¤0K001  A     ¤0P001,¤0M001-2&1                   7    633  A658666    658   666.
0035 *             S                                         1    640  S                    .
0036 *     ¤0L001  BCE   ¤0J001,¤0N001,&                     8    641  B608660&   608   660.
0037 *             B     ¤0K001                              4    649  B633       633      .
0038 *             DCW   @1@                                 1    653                       .
0039 *     ¤0P001  DCW   #5                                       658                       .
0040 *     ¤0N001  DCW   #2                                       660                       .
0041 *     ¤0M001  DCW   #00007                                   667                       .
0042 *     ¤0Q001  EQU   *                                        667                       .
0043 *     ¤0R001  EQU   *&1                                      668                       .
0044 *             ZA    ¤0M001,ACCUM                        7    668  &667U11    667   1411.
0045 0026          MLCWA MASK,AREA                           7    675  LT90U04    1390  1404.
0046 0027          MCE   ACCUM,AREA                          7    682  EU11U04    1411  1404.
0047 0028          SW    AREA-8                              4    689  ,T96       1396       .
0048 0029          MLC   AREA,232                            7    693  MU04232    1404  232.
0049 0030          A     ACCUM,TOT2                          7    700  AU11U27    1411  1427.
0050 0031            DIVIDFIFTH,2,0,ACCUM,7,2,ACCUM,7,2                **MACRO**            .
0051 *             ZA    FIFTH,¤0Q002                        7    707  &U47848    1447  848.
0052 *             A     @0@,¤0Q002                          7    714  AU63848    1463  848.
0053 *             S     ¤1L002                              4    721  S865       865      .
0054 *             ZA    ACCUM,¤1L002-00001                  7    725  &U11864    1411  864.
0055 *             ZA    ¤1L002                              4    732  &865       865      .
```

Figure 35. Sample Program (Part 2 of 6)

```
SPL01                    SALARY TABLE COMPUTATIONS                              PAGE    2

SEQ  PGLIN LABEL   OPCD  OPERAND                          SFX CT   LOCN  INSTRCTN A-ADD B-ADD .FLAGS

0056 *             A     @0@,¤1L002                         7   736  AU63865  1463   865 .
0057 *             S     ¤1K002                             4   743  S856      856       .
0058 *     ¤0K002  MLCWA ¤0P002,¤0R002-1                    7   747  L839873   839   873 .
0059 *             ZA    ¤0Q002,¤1J002-1                    7   754  &848882   848   882 .
0060 *             ZA    ¤1J002,¤0Q002                      7   761  &883848   883   848 .
0061 *             MLCWA                                    1   768  L
0062 *             C     ¤0Q002,¤1L002                      7   769  C848865   848   865 .
0063 *             BH    ¤0K002                             5   776  B747U     747       .
0064 *     ¤0L002  C     ¤0Q002,¤1L002                      7   781  C848865   848   865 .
0065 *             BL    ¤0M002                             5   788  B805T     805       .
0066 *             S     ¤0Q002,¤1L002                      7   793  S848865   848   865 .
0067 *             A                                        1   800  A                   .
0068 *             B     ¤0L002                             4   801  B781      781       .
0069 *     ¤0M002  BCE   ¤1N002,¤0P002,1                    8   805  B8848391  884   839 .
0070 *             MLCWA ¤0P002-1,¤0P002                    7   813  L838839   838   839 .
0071 *             ZA    ¤0Q002-1,¤0Q002                    7   820  &847848   847   848 .
0072 *             B     ¤0L002                             4   827  B781      781       .
0073 *             DCW   #00008                                838                       .
0074 *     ¤0P002  DCW   @1@                                1   839                       .
0075 *     ¤0Q002  DCW   #00009                                848                       .
0076 *     ¤1K002  DCW   #00008                                856                       .
0077 *     ¤1L002  DCW   #00009                                865                       .
0078 *             DCW   #00008                                873                       .
0079 *     ¤0R002  DC    @0@                                1   874                       .
0080 *             DCW   #00008                                882                       .
0081 *     ¤1J002  DC    @0@                                1   883                       .
0082 *     ¤1N002  EQU   *&1                                   884                       .
0083 *             A     @5@,¤1K002                         7   884  AU64856  1464   856 .
0084 *             ZA    ¤1K002-1,ACCUM                     7   891  &855U11   855  1411 .
0085 0032          MLCWA MASK,AREA                          7   898  LT90U04  1390  1404 .
0086 0033          MCE   ACCUM,AREA                         7   905  EU11U04  1411  1404 .
0087 0034          SW    AREA-8                             4   912  ,T96     1396       .
0088 0035          MLC   AREA,247                           7   916  MU04247  1404   247 .
0089 0036          A     ACCUM,TOT3                         7   923  AU11U35  1411  1435 .
0090 0037            DIVIDFORTY,2,0,ACCUM,7,2,ACCUM,7,2                **MACRO**         .
0091 *             ZA    FORTY,¤0Q003                       7   930  &U49#71  1449  1071 .
0092 *             A     @0@,¤0Q003                         7   937  AU63#71  1463  1071 .
0093 *             S     ¤1L003                             4   944  S#88     1088       .
0094 *             ZA    ACCUM,¤1L003-00001                 7   948  &U11#87  1411  1087 .
0095 *             ZA    ¤1L003                             4   955  &#88     1088       .
0096 *             A     @0@,¤1L003                         7   959  AU63#88  1463  1088 .
0097 *             S     ¤1K003                             4   966  S#79     1079       .
0098 *     ¤0K003  MLCWA ¤0P003,¤0R003-1                    7   970  L#62#96  1062  1096 .
0099 *             ZA    ¤0Q003,¤1J003-1                    7   977  &#71/05  1071  1105 .
0100 *             ZA    ¤1J003,¤0Q003                      7   984  &/06#71  1106  1071 .
0101 *             MLCWA                                    1   991  L                   .
0102 *             C     ¤0Q003,¤1L003                      7   992  C#71#88  1071  1088 .
0103 *             BH    ¤0K003                             5   999  B970U     970       .
0104 *     ¤0L003  C     ¤0Q003,¤1L003                      7  1004  C#71#88  1071  1088 .
0105 *             BL    ¤0M003                             5  1011  B#28T    1028       .
0106 *             S     ¤0Q003,¤1L003                      7  1016  S#71#88  1071  1088 .
0107 *             A                                        1  1023  A                   .
0108 *             B     ¤0L003                             4  1024  B#04     1004       .
0109 *     ¤0M003  BCE   ¤1N003,¤0P003,1                    8  1028  B/07#621  1107  1062 .
0110 *             MLCWA ¤0P003-1,¤0P003                    7  1036  L#61#62   1061  1062 .
```

Figure 35. Sample Program (Part 3 of 6)

```
SPL01                    SALARY TABLE COMPUTATIONS                              PAGE    3

SEQ  PGLIN LABEL    OPCD  OPERAND                                  SFX CT  LOCN  INSTRCTN A-ADD B-ADD .FLAGS

0111 *             ZA    ¤0Q003-1,¤0Q003                            7  1043  &#70#71  1070  1071  .
0112 *             B     ¤0L003                                     4  1050  B#04     1004        .
0113 *             DCW   #00008                                        1061                      .
0114 *     ¤0P003  DCW   @1@                                        1  1062                      .
0115 *     ¤0Q003  DCW   #00009                                        1071                      .
0116 *     ¤1K003  DCW   #00008                                        1079                      .
0117 *     ¤1L003  DCW   #00009                                        1088                      .
0118 *             DCW   #00008                                        1096                      .
0119 *     ¤0R003  DC    @0@                                        1  1097                      .
0120 *             DCW   #00008                                        1105                      .
0121 *     ¤1J003  DC    @0@                                        1  1106                      .
0122 *     ¤1N003  EQU   *&1                                           1107                      .
0123 *             A     @5@,¤1K003                                 7  1107  AU64#79  1464  1079  .
0124 *             ZA    ¤1K003-1,ACCUM                             7  1114  &#78U11  1078  1411  .
0125 0038          MLCWA MASK,AREA                                  7  1121  LT90U04  1390  1404  .
0126 0039          MCE   ACCUM,AREA                                 7  1128  EU11U04  1411  1404  .
0127 0040          SW    AREA-8                                     4  1135  ,T96     1396        .
0128 0041          A     ACCUM,TOT4                                 7  1139  AU11U43  1411  1443  .
0129 0042          MLC   AREA,262                                   7  1146  MU04262  1404   262  .
0130 0043          W                                                1  1153  2                    .
0131 0044          CS    299                                        4  1154  /299      299        .
0132 0045          A     @25@,MTH-2                                 7  1158  AU66T93  1466  1393  .
0133 0046          C     MTH-2,@901@                                7  1165  CT93U69  1393  1469  .
0134 0047          BH    START                                      5  1172  B553U     553        .
0135 0048          W                                                1  1177  2                    .
0136 0049          W                                                1  1178  2                    .
0137 0050          MLC   @FIRST LINE IS COMPUTED TOTALS@,229        7  1179  MU98229  1498   229  .
0138 0051          W                                                1  1186  2                    .
0139 0052          CS    299                                        4  1187  /299      299        .
0140 0053          MLC   @SECOND LINE IS CORRECT TOTALS@,229        7  1191  MV27229  1527   229  .
0141 0054          W                                                1  1198  2                    .
0142 0055          CS    299                                        4  1199  /299      299        .
0143 0056          W                                                1  1203  2                    .
0144 0057          MLCWA MASK,AREA                                  7  1204  LT90U04  1390  1404  .
0145 0058          MCE   TOT1,AREA                                  7  1211  EU19U04  1419  1404  .
0146 0059          SW    AREA-8                                     4  1218  ,T96     1396        .
0147 0060          MLC   AREA,216                                   7  1222  MU04216  1404   216  .
0148 0061          MLCWA MASK,AREA                                  7  1229  LT90U04  1390  1404  .
0149 0062          MCE   TOT2,AREA                                  7  1236  EU27U04  1427  1404  .
0150 0063          SW    AREA-8                                     4  1243  ,T96     1396        .
0151 0064          MLC   AREA,232                                   7  1247  MU04232  1404   232  .
0152 0065          MLCWA MASK,AREA                                  7  1254  LT90U04  1390  1404  .
0153 0066          MCE   TOT3,AREA                                  7  1261  EU35U04  1435  1404  .
0154 0067          SW    AREA-8                                     4  1268  ,T96     1396        .
0155 0068          MLC   AREA,247                                   7  1272  MU04247  1404   247  .
0156 0069          MLCWA MASK,AREA                                  7  1279  LT90U04  1390  1404  .
0157 0070          MCE   TOT4,AREA                                  7  1286  EU43U04  1443  1404  .
0158 0071          SW    AREA-8                                     4  1293  ,T96     1396        .
0159 0072          MLC   AREA,262                                   7  1297  MU04262  1404   262  .
0160 0073          W                                                1  1304  2                    .
0161 0074          CS    299                                        4  1305  /299      299        .
0162 0075          MLC   @13650.00@,216                             7  1309  MV35216  1535   216  .
0163 0076          MLC   @163800.00@,232                           7  1316  MV44232  1544   232  .
0164 0077          MLC   @3150.00@,247                              7  1323  MV51247  1551   247  .
0165 0078          MLC   @78.75@,262                                7  1330  MV56262  1556   262  .
```

Figure 35. Sample Program (Part 4 of 6)

```
     SPL01                    SALARY TABLE COMPUTATIONS                                    PAGE    4

SEQ  PGLIN LABEL   OPCD  OPERAND                                  SFX CT   LOCN  INSTRCTN A-ADD B-ADD  .FLAGS

0166 0079          W                                               1     1337  2
0167 0080          CS    299                                       4     1338  /299       299
0168 0081          W                                               1     1342  2
0169 0082          W                                               1     1343  2
0170 0083          MLC   @END OF TABLE@,241                        7     1344  MV68241   1568   241
0171 0084          W                                               1     1351  2
0172 0085            SYSCL                                                     **MACRO**
0173 *      ***  RETURN CONTROL TO SYSTEM  ***
0174 *             B     3928                                      4     1352  BI28       3928
0175 *             NOP                                             1     1356  N
0176 0086  MONTH   DCW   @MONTHLY@                                 7     1363
0177 0087  YEAR          @YEARLY@                                  6     1369
0178 0088  WEEK          @WEEKLY@                                  6     1375
0179 0089  HOUR          @HOURLY@                                  6     1381
0180 0090  MASK          @     0.  @                               9     1390
0181 0091  MTH           @40000@                                   5     1395
0182 0092  AREA          #9                                              1404
0183 0093  ACCUM         #7                                              1411
0184 0094  TOT1          00000000                                  8     1419
0185 0095  TOT2          00000000                                  8     1427
0186 0096  TOT3          00000000                                  8     1435
0187 0097  TOT4          00000000                                  8     1443
0188 0098  TWLV  DCW     @12@                                      2     1445
0189 0099  FIFTW DCW     @52@                                      2     1447
0190 0100  FORTY DCW     @40@                                      2     1449
0191 0101          END   BEG                                                 500        500
0192               LTRL  @SALARY TABLE@                           12     1461
0193               LTRL  &0                                        1     1462
0194               LTRL  @0@                                       1     1463
0195               LTRL  @5@                                       1     1464
0196               LTRL  @25@                                      2     1466
0197               LTRL  @901@                                     3     1469
0198               LTRL  @FIRST LINE IS COMPUTED TOTALS@          29     1498
0199               LTRL  @SECOND LINE IS CORRECT TOTALS@          29     1527
0200               LTRL  @13650.00@                                8     1535
0201               LTRL  @163800.00@                               9     1544
0202               LTRL  @3150.00@                                 7     1551
0203               LTRL  @78.75@                                   5     1556
0204               LTRL  @END OF TABLE@                           12     1568
                           END OF LISTING                                 NO SEQUENCE ERRORS
```

```
CORE LOAD HEADER-   SALARY TABLE COMPUTATIONS                    , ID-SPL01
CORE LOAD OUTPUT COMPLETE ON 1311 UNIT 0, START 012300, END 012318
```

Figure 35. Sample Program (Part 5 of 6)

```
                          SALARY TABLE

        MONTHLY          YEARLY          WEEKLY          HOURLY

          400.00          4800.00           92.31            2.31
          425.00          5100.00           98.08            2.45
          450.00          5400.00          103.85            2.60
          475.00          5700.00          109.62            2.74
          500.00          6000.00          115.38            2.88
          525.00          6300.00          121.15            3.03
          550.00          6600.00          126.92            3.17
          575.00          6900.00          132.69            3.32
          600.00          7200.00          138.46            3.46
          625.00          7500.00          144.23            3.61
          650.00          7800.00          150.00            3.75
          675.00          8100.00          155.77            3.89
          700.00          8400.00          161.54            4.04
          725.00          8700.00          167.31            4.18
          750.00          9000.00          173.08            4.33
          775.00          9300.00          178.85            4.47
          800.00          9600.00          184.62            4.62
          825.00          9900.00          190.38            4.76
          850.00         10200.00          196.15            4.90
          875.00         10500.00          201.92            5.05
          900.00         10800.00          207.69            5.19


FIRST LINE IS COMPUTED TOTALS
SECOND LINE IS CORRECT TOTALS

        13650.00        163800.00         3150.00           78.75
        13650.00        163800.00         3150.00           78.75


                          END OF TABLE
                               HALT
```

Figure 35. Sample Program (Part 6 of 6)

Index

READER'S COMMENT FORM

Autocoder (on Disk) Specifications and Operating Procedures IBM 1401, 1440, 1460
Form C24-3259-3

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. All comments will be handled on a non-confidential basis.

	Yes	No
• Does this publication meet your needs?	☐	☐
• Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

- What is your occupation? _____.
- How do you use this publication?

As an introduction to the subject?	☐	As an instructor in a class?	☐
For advanced knowledge of the subject?	☐	As a student in a class?	☐
For information about operating procedures?	☐	As a reference manual?	☐

Other _____.

- Please give specific page and line references with your comments when appropriate. If you wish a reply, be sure to include your name and address.

COMMENTS:

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

fold fold

FIRST CLASS
PERMIT NO. 387
ROCHESTER, MINN.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Systems Development Division
Development Laboratory
Rochester, Minnesota 55901

Attention: Programming Publications, Dept. 425

fold fold

IBM 1440, 1401, 1460 Printed in U. S. A. C24-3259-3

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601

IBM 1440, 1401, 1460 Printed in U. S. A. C24-3259-3

IBM®
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601