File No. GENL-24
Form C24-3242-2

IBM Systems Reference Library

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

*Program Number 1440-CB-073*

This publication contains the operating procedures for building the COBOL system using IBM 1311 and 1301 Disk Storage, modifying the COBOL system, and assembling a machine-language object program from a source program written in the COBOL language. A description of the phases that make up the COBOL system, a list of halts and messages, and a sample program are included.

# Preface

This publication contains the program specifications and operating procedures for the COBOL (on disk) programming system for IBM 1401, 1440, and 1460. In this publication, the term COBOL system refers to *1401/1440/1460 COBOL (on Disk)*, program number 1440-CB-073. The language specifications for the COBOL system are contained in the Systems Reference Library publication *COBOL (on Disk) Specifications for IBM 1401, 1440, and 1460,* Form C24-3235.

This publication is divided into two major sections; program specifications and operating procedures. The program specifications describe the COBOL system. Included in the section are such topics as a description of the System Control Program (the controlling element of the COBOL system), a description of the COBOL compiler, and a detailed description of the results of system operations. Although this section is directed primarily at 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 a COBOL 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 at the machine operator, it is recommended that the programmer review the content of the entire 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 COBOL programming. It is recommended that these publications be available to the user for reference purposes.

*COBOL General Information Manual,* Form F28-8053.

*COBOL (on Disk) Specifications for IBM 1401, 1440, and 1460,* Form C24-3235.

In addition, it is recommended that the user be familiar with the following Systems Reference Library Autocoder publications.

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

*Autocoder (on Disk) Program Specifications and Operating Procedures for IBM 1401, 1440, and 1460,* Form C24-3259.

If the user's system is IBM 1401 or IBM 1460, it is recommended that the user have aeeess to the following Systems Reference Library publications.

*Input/Output Control System (on Disk) Specifications for IBM 1401 and 1460,* Form C24-1489.

*Input/Output Control System (on Disk) Operating Procedures for IBM 1401 and 1460,* Form C24-3298.

*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.

If the user's system is IBM 1440, it is recommended that the user have access to the following Systems Reference Library publications.

*Input/Output Control System Specifications for IBM 1440,* Form C24-3011.

*Input/Output Control System Operating Procedures for IBM 1440,* Form C24-3299.

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

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

*Major Revision, February 1965*

This edition, C24-3242-2, is a major revision of and obsoletes C24-3242-1.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.
Address comments concerning the content of this publication to IBM Product Publications, Endicott, New York 13764.

# Contents


**Program Specifications** ..... 5

Definition of Key Terms ..... 5
Machine Requirements ..... 5

**The COBOL System** ..... 6

System Control Program ..... 6
Logical Files ..... 6
  Residence File ..... 7
  Operation Files ..... 7
  External Files ..... 7
  Internal Files ..... 7
Control Cards ..... 7
  RUN Card ..... 7
  ASGN Cards ..... 8
  UPDAT Card ..... 8
  NOTE Card ..... 8
  PAUSE Card ..... 9
  HALT Card ..... 9
COBOL Compiler ..... 9
Autocoder Assembler ..... 9
IOCS ..... 9
COBOL Macros ..... 9
COBOL Compiler Output ..... 13
COBOL Diagnostic Messages ..... 13

**Operating Procedures** ..... 20

Jobs ..... 20
Preparing Processor Jobs ..... 21
  COBOL RUN ..... 21
  COBOL RUN THRU AUTOCODER ..... 21
  COBOL RUN THRU OUTPUT ..... 22
  COBOL RUN THRU EXECUTION ..... 23
Changing File Assignments ..... 25
  Preparing ASGN Cards ..... 25
  Using ASGN Cards ..... 28
Batched Files ..... 28

**Performing Jobs** ..... 28

Preparing a Stack ..... 28
Running a Stack ..... 29
Loading Object Programs ..... 30
Halts and Messages ..... 30

**Building and Updating a COBOL System** ..... 36

COBOL-System Deck Description and Preparation ..... 36
  Marking Program ..... 37
  System Control Modification ..... 38
  Write File-Protected Addresses ..... 38
  COBOL Update ..... 38
  COBOL Macros ..... 38
  COBOL Sample Program ..... 38
Building a COBOL System ..... 38
  System Control Modification ..... 39
  Write File-Protected Addresses ..... 40
  COBOL Update and COBOL Macros ..... 41
  COBOL Sample Program ..... 42

**Updating a COBOL System** ..... 42

**Appendix I—Control Card Formats** ..... 43

**Appendix II—Phase Descriptions** ..... 46

**Appendx III—COBOL Macros** ..... 51

**Appendix IV—Sample Program** ..... 52

**Index** ..... 60

This publication describes the program specifications and operating procedures used to assemble a machine-language program from a source program written in IBM 1401, 1440, and 1460 COBOL language. It also describes the operating procedures for building and modifying the COBOL system on a disk unit.

This system resides on a file-protected area of a disk-storage unit. By the use of the System Control Program, the controlling element of the COBOL system, it is possible to stack the input to and output from a series of tasks. Further, the System Control Program allows the user to assign input/output devices for a defined set of logical files.

The similarity between COBOL and ordinary business English provides programmers with a convenient method for writing source programs. Source program statements are translated into machine language by the COBOL system. This permits the programmer to direct his attention primarily toward the solution of the problem, rather than toward the specific method of implementing the solution on the machine.

### 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.*

*Batched Files.* Logical files whose contents represent one or more sequential sets of input to or output from the COBOL system.

*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 COBOL system.

*Logical Files.* Input/output devices and/or areas used by the COBOL 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 compiling, 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

To process a COBOL source program, the following minimum machine configurations are specified.

An IBM 1401 System with:

4,000 positions of core storage
Advanced Programming Feature
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
Indexing and Store Address Register Feature
One IBM 1311 Disk Storage Drive
One IBM 1442 Card Reader
One IBM 1443 Printer

An IBM 1440 System with:

8,000 positions of core storage
Indexing and Store Address Register Feature
One IBM 1301 Disk Storage
One IBM 1442 Card Reader
One IBM 1443 Printer

An IBM 1460 System with:

8,000 positions of core storage
One IBM 1311 Disk Storage Drive, or one IBM 1301 Disk Storage
One IBM 1311 Disk Storage Drive
One IBM 1402 Card Read-Punch
One IBM 1403 Printer

The system on which the object program is to be executed must have:

1. A card reader or a disk file to load the object program resulting from the Output processor.
2. Sufficient core storage to contain the program generated by the COBOL compiler. If the object program requires more than the available core-storage capacity, either the program must be executed in sec-

tions (overlays) or the job must be divided into multiple runs.

3. The input and output devices defined in the FILE-CONTROL paragraph or the SPECIAL-NAMES paragraph.
4. Sense switches when they are referred to in the SPECIAL-NAMES paragraph.

The COBOL system can use the following devices, if available:

IBM 1444 Card Punch
IBM 1447 Console without a buffer feature.

## The COBOL System

The IBM 1401, 1440, and 1460 COBOL system built by the user contains a System Control Program, the COBOL compiler, the Autocoder assembler, Input/Output Control System (IOCS), and COBOL macro instructions.

*System Control Program.* The main function of the System Control Program is to analyze control-card information, determine the specific control mode, and transfer program control to the appropriate portion of the system.

*COBOL Compiler.* The COBOL compiler portion of the system operates under the control of the System Control Program. The compiler translates the source program, written in the COBOL language, into Autocoder-IOCS symbolic statements.

*Autocoder Assembler.* The Autocoder assembler portion of the COBOL system operates on the results of the COBOL compiler. The Autocoder library must contain IOCS and the COBOL macro instructions. For a complete description of the Autocoder assembler, see the Systems Reference Library publication, *Autocoder (on Disk) Program Specifications and Operating Procedures for IBM 1401, 1440, and 1460,* Form C24-3259.

*IOCS.* Standard input/output macro instructions are used by the COBOL system for all COBOL files. A complete description of IOCS is in either *Input/Output Control System (on Disk) Operating Procedures for IBM 1401 and 1460,* Form C24-3298, or *Input/Output Control System Operating Procedures for IBM 1440,* Form C24-3299.

*COBOL Macros.* COBOL macros, a special set of macro instructions, are used by the COBOL system to set permanent switches during the assembly for the use of other macros, to generate the COBOL object-time subroutines, and to generate a limited set of inline expansions.

## 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, a source program is coded in the COBOL language, and the user specifies the end result of processing to be a machine-language object program. This would require that processing be performed by the COBOL compiler, the Autocoder processor, and the Output processor. The control card says in effect that the source program is coded in COBOL, and that processing is to run through the Output processor. The System Control Program reads the control card and calls in the COBOL compiler. Processing takes place, and at completion, control reverts to the System Control Program. The Autocoder processor is called, processing takes place, and at completion, control reverts to the System Control Program. The Output processor is called, processing takes place, and at 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.

### Logical Files

A set of logical files, defined by the COBOL 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.

### Residence File

*SYSTEM File.* The SYSTEM file contains the System Control Program, the COBOL Compiler Program, and the Autocoder Assembler Program. It is assigned to a fixed area in a 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

*WORK Files.* The WORK files (WORK1 through WORK5) are mass-storage files that contain intermediate results of the processors. The WORK files can be assigned to any available area in disk storage.

## Control Cards

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

RUN
ASGN
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 COBOL system that are to be selected by the System Control Program. A RUN card is required for each job to be performed. The general format of the RUN card is:

```
COBOL   RUN   [ THRU { AUTOCODER }  ]
              [      { OUTPUT    }  ]
              [      { EXECUTION }  ]
```

If the optional part of the RUN card is omitted (THRU AUTOCODER, THRU OUTPUT, or 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:

```
COBOL RUN
COBOL RUN THRU AUTOCODER
COBOL RUN THRU OUTPUT
COBOL RUN THRU EXECUTION
```

### 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:

*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 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 is when a logical file is to be omitted. (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 or another ASGN card is sensed for that particular file. For example, an ASGN card that specifies the INPUT file to be assigned to READER 2 causes the assumed assignment, READER 1, to be altered. The System Control Program will select READER 2 during a single stack until another ASGN card for the INPUT file is encountered.

### UPDAT Card

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

{ *processor-name* / SYSTEM } UPDAT *phase-name,* { ALL / DELETE / HEADER / INSERT / PATCH }

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 compilations be placed on the OUTPUT file located on disk-drive 2. A NOTE card could be used, at the completion of processing the series of jobs, 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*

## COBOL Compiler

The COBOL compiler, operating under the control of the System Control Program, translates source programs written in the COBOL language into Autocoder-IOCS statements.

In order that this translating process be accomplished, the compiler is divided into ten logical segments, each of which is made up of phases. They are:

| *Phases* | *Function* |
|---|---|
| A | Syntax Scan |
| B | Name Compression and Diagnostics |
| C | Data Prescan |
| D | Data Scan |
| E | IOCS |
| F | Procedure Prescan |
| G | Procedure Scan |
| H | Data-Descriptions Merge |
| I | General to Particular Macros |
| J | Macro Expander |

A complete description of the phases that comprise the COBOL compiler is contained in *Appendix II.*

## Autocoder Assembler

The Autocoder assembler contained in the COBOL system is the IBM 1401, 1440, and 1460 Autocoder system that is under control of the System Control Program. The Autocoder system is described in detail in the Systems Reference Library publication *Autocoder (on Disk) Program Specifications and Operating Procedures for IBM 1401, 1440, and 1460,* Form C24-3259.

## IOCS

Standard macro instructions of the Input/Output Control System (IOCS) are used by the COBOL system. Detailed information pertaining to the macro system can be found in the Systems Reference Library publication *Input/Output Control System (on Disk) Operating Procedures for IBM 1401 and 1460,* Form C24-3298, or *Input/Output Control System Operating Procedures for IBM 1440,* Form C24-3299.

## COBOL Macros

In addition to the IOCS macro instructions, the system includes twenty-five special COBOL macros. *Appendix III* shows the macros, the names of the subroutines that are called by using the macros, and the reason for calling the routines.

### *ACEPT*

The ACEPT macro generates the *accept* subroutine. The low-order position and the length of a field is specified. The accept subroutine reads the appropriate number of characters to fill this field. The linkage used for the accept subroutine is:

```
B     ZAX
DCW   001      (Reader)
      002      (Console Printer)
DCW   004      (Pocket-1402)
      002
      000
DCW   001      (Unit-1442)
      002
DCW   NAME1
DCW   @0@      (Not Subscripted)
      @1@      (Subscripted)
DCW   009      (Name Size)
```

### *ALCOM*

The ALCOM macro generates the *alphabetic-compare* subroutine. This subroutine is used when two alphabetic records with subfields are to be compared. The linkage used for the alphabetic-compare subroutine is:

```
B     YAQ
DCW   NAME1
DCW   n        (Not used)
DCW   nnn      (Length of NAME1)
DCW   NAME2
DCW   n        (Not used)
DCW   nnn      (Length of NAME2)
DCW   @bSS@    (NAME1 ≠ NAME2)
      @b//@    (NAME1 = NAME2)
      @bTS@    (NAME1 > NAME2)
      @bUS@    (NAME1 < NAME2)
      @bTT@    (NAME1 ≥ NAME2)
      @bUU@    (NAME1 ≤ NAME2)
DCW   BRANCH LABEL
```

### *DIVDE*

The DIVDE macro generates the *divide* subroutine. The subroutine is used as a substitute for the divide operation code. The linkage used for the divide subroutine is:

```
B     DIV
DCW   NAME1
DCW   NAME2
```

### *DSPLY*

The DSPLY macro generates the *display* subroutine. The number of fields, the input device, the low-order position of each field, and the length of each field are specified. The display subroutine packs these fields sequentially in a buffer area, and subsequently outputs

them on the specified display device (the card punch, the printer, or the console printer). The linkage used for the display subroutine is:

```
B      ZDY
DCW    000        (Punch)
       001        (Printer)
       002        (Console Printer)
DCW    004        (Pocket-1402)
       008
       000
DCW    001        (Unit-1442)
       002
DCW    nnn        (Number of names to be displayed)
DCW    N          (Name of field)
DCW    @0@        (Not subscripted)
       @1@        (Subscripted)
DCW    nnn        (Length of Name)
        .
        .
        .
DCW    Nn
DCW    @0@n
       @1@n
DCW    nnn
```

### EDIT1

The EDIT1 macro generates the *editing* subroutine. This subroutine is used when editing which is not included within the 1401, 1440, or 1460 Central Processing Unit is used by the COBOL program. Editing features that are not handled by the central processing units are: COBOL zeros, floating plus or minus signs, single plus sign, and DB. When the editing function is completed, the subroutine returns to the location following the parameters of this subroutine. The linkage used for the editing subroutine is:

```
B      ZET
DCW    NAME1      (Sending field)
DCW    0          (Not subscripted)
       1          (Subscripted)
DCW    NAME2      (Label of COBOL mask)
       000        (No COBOL mask)
DCW    @DbD@      (Replace CR by DB)
       @nnn@      (Position of plus sign)
       @KbK@      (Neither editing feature desired)
DCW    @+b+@      (Floating plus sign)
       @−b−@      (Floating minus sign)
       @KbK@      (Neither editing feature desired)
DCW    NAME2      (Receiving field)
DCW    0          (Not subscripted)
       1          (Subscripted)
DCW    nnn        L1 = Length of edited field
DCW    nnn        L2 = Length of field to be edited
```

### EXPIN

The EXPIN macro generates the *exponentiation* subroutine that is used when a number is exponentiated to an integer power. The linkage used for the exponentiation subroutine is:

```
B      ZFZ
DCW    NAME1      (Name of base)
DCW    nnn        (Number of decimals if NAME1 = GT)*
DCW    n          (Subscript indicator)
DCW    nnn        (Number of decimals in NAME1)
DCW    NAME2      (Name of power)
DCW    nnn        (Number of decimals if NAME2 = GT)*
DCW    n          (Subscript indicator)
DCW    nnn        (Number of decimals in NAME2)
DCW    NAME3      (Name of intermediate result field)
DCW    SIZE1      (Length of field 1)
DCW    SIZE2      (Length of field 2)
```

*GT is equivalent to generated temporary result field.

### EXPNI

The EXPNI macro generates the *exponentiation* subroutine that is used when a number is exponentiated to a decimal power. The linkage used for the exponentiation subroutine is:

```
B      ZXZ
DCW    NAME1      (Name of base)
DCW    nnn        (Number of decimals if NAME1 = GT)*
DCW    n          (Subscript indicator)
DCW    nnn        (Number of decimals in NAME1)
DCW    NAME2      (Name of power)
DCW    nnn        (Number of decimals if NAME2 = GT)*
DCW    n          (Subscript indicator)
DCW    nnn        (Number of decimals in NAME2)
DCW    NAME3      (Name of intermediate result field)
DCW    SIZE1      (Length of field 1)
DCW    SIZE2      (Length of field 2)
```

*GT is equivalent to generated temporary result field.

### FGCOM

The FGCOM macro generates the *compare-figcon* subroutine. This subroutine is used when a record with subfields is to be compared to a figurative constant (high value, low value, quote, and ALL alpha literal). The linkage used for the compare-figcon subroutine is:

```
B      YCL
DCW    NAME1
DCW    n                (Not used)
DCW    nnn              (Length of NAME1)
DCW    NAME2
DCW    n                (Not used)
DCW    nnn              (Length of NAME2)
DCW    @bSS@            (NAME1 ≠ NAME2)
       @b//@            (NAME1 = NAME2)
       @bTS@            (NAME1 > NAME2)
       @bUS@            (NAME1 < NAME2)
       @bTT@            (NAME1 ≥ NAME2)
       @bUU@            (NAME1 ≤ NAME2)
DCW    BRANCH LABEL
DCW    000              (Neither figcon)
       001              (NAME1 figcon)
       002              (NAME2 figcon)
```

### GOTOD

The GOTOD macro generates the *go-to-depending* subroutine. This subroutine tests the value of data-name (NAME1). If the value of data-name exceeds the number of branch levels given, or if it is equal to zero, the subroutine returns to the location following the parameters of this subroutine. Otherwise, the subroutine branches to the 1st, 2nd, . . ., nth branch label address if the value of data-name is 1, 2, . . ., n. The linkage used for the go-to-depending subroutine is:

```
B      ZGP
DCW    nnn          (Number of branch labels)
DCW    NAME1
DCW    nnn
DCW    0            (Not subscripted)
       1            (Subscripted)
DCW    LABEL1       (Branch labels)
         .
         .
         .
DCW    LABELn
```

### IFALP

The IFALP macro generates the *if-alphabetic* subroutine. This subroutine is used when a field is to be tested to see whether it is alphabetic. If both the result of the test and the operator are true, the subroutine branches to the label address given as parameter 4. If either fails, the subroutine returns to the location following the parameters of this subroutine. The linkage used for the if-alphabetic subroutine is:

```
B      YIP
DCW    NAME1
DCW    0                (Not subscripted)
       1                (Subscripted)
DCW    nnn              (Length of NAME1)
DCW    BRANCH LEVEL
DCW    @$10@            (Alphabetic)
       @b10@            (Not alphabetic)
```

### IFNUM

The IFNUM macro generates the *if-numeric* subroutine. This subroutine is used when a field is to be tested to see whether it is numeric. If the result of this test agrees with the operator, the subroutine branches to the label address given as parameter 4. If the result does not agree with the operator, the subroutine returns to the location following the parameters of the routine. The linkage used for the if-numeric subroutine is:

```
B      YIN
DCW    NAME1
DCW    0                (Not subscripted)
       1                (Subscripted)
DCW    nnn              (Size of NAME1)
DCW    BRANCH LABEL
DCW    @$16@            (Numeric)
       @b16@            (Not numeric)
```

### INDIX

The INDIX macro generates the *subscript-index* subroutine. This subroutine is used by all COBOL subroutines except DISPLAY. Its function is to test the index indicator parameter and substitute the address of the required index register in place of the switch. The linkage used for the subscript-index subroutine is:

```
B      ZSP
DCW    001      (X1)
       002      (X2)
       003      (X3)
```

### MULTY

The MULTY macro generates the *multiply* subroutine. This subroutine is used as a substitute for the multiply-operation code. The linkage used for the multiply subroutine is:

```
B      MULTY
DCW    NAME1
DCW    NAME2
```

### MVALL

The MVALL macro generates the *move-all* subroutine. This subroutine is used when a record with subfields is to be filled with a figurative constant, for example, ALL@ABC@. The figurative constant is moved into the record from left to right until the record is filled. When the record is filled, the subroutine returns to the location following the parameters of the subroutine. The linkage used for the move-all subroutine is:

```
B      ZML
DCW    NAME1    (Receiving field)
DCW    n        (Not used)
DCW    nnn      (Length of the field)
DCW    NAME2    (Figcon)
DCW    nnn      (Length of figcon)
```

### *MVFTR*

The MVFTR macro generates the *move-field-to-record* subroutine. This subroutine is used when the record has subfields. The field is moved to the record area and justified according to parameter 7. When the field has been moved to the record, the subroutine returns to the location following the parameters of this subroutine. The linkage used for the move-field-to-record subroutine is:

```
B     ZMR
DCW NAME1     (Field or record name)
DCW n         (Not used)
DCW nnn       (Length of field or record)
DCW NAME2     (Field or record name)
DCW n         (Not used)
DCW nnn       (Length of field or record)
DCW @bbR@     (Right)
    @bbL@     (Left)
```

### *SPLIT*

The SPLIT macro generates the *stop-literal* subroutine. This subroutine displays the specified literal or the address of the specified literal in the address register or on the console printer. The linkage used for the stop-literal subroutine is:

```
B     SLT
DCW NAME1
```

### *SUBS1, SUBS2, SUBS3*

The SUBS1, SUBS2, and SUBS3 macros generate the *subscripted* subroutines. These three subroutines are used independently to compute an address for a data-name that has been subscripted. The address is computed according to the following equations, depending upon whether the data-name has 1, 2, or 3 subscripted variables.

$$CA = B + (V_1 - 1)\,S_1$$
$$CA = B + (V_1 - 1)\,S_1 + (V_2 - 1)\,S_2$$
$$CA = B + (V_1 - 1)\,S_1 + (V_2 - 1)\,S_2 + (V_3 - 1)\,S_3$$

where: CA is the Computed Address
B is the Basic Address of the data-name
$V_1$, $V_2$, $V_3$ are the subscripted variable values.
$S_1$, $S_2$, $S_3$, are the sizes of the basic areas.

After the address has been computed, the subroutines return to the location following the parameters of these subroutines. The linkage used for the subscript-1 subroutine is:

```
B     XXJ
DCW NAME1
DCW nnn       (Size of table element)
DCW NAME2     (Address of subscripted variable)
DCW LABEL     (Address of field where computed
               address is to be stored)
      or
      001     (X1)
      002     (X2)
      003     (X3)
DCW nnn       (Size of NAME1)
```

The linkage used for the subscript-2 subroutine is:

```
B     XXK
DCW NAME1
DCW nnn       (Size of table element 1)
DCW nnn       (Size of table element 2)
DCW NAME2     (Address of subscripted variable 1)
DCW NAME3     (Address of subscripted variable 2)
DCW LABEL     (Address of field where computed
               address is to be stored)
      or
      001     (X1)
      002     (X2)
      003     (X3)
DCW nnn       (Size of NAME1)
```

The linkage used for the subscript-3 subroutine is:

```
B     XXL
DCW NAME1
DCW nnn       (Size of table element 1)
DCW nnn       (Size of table element 2)
DCW nnn       (Size of table element 3)
DCW NAME2     (Address of subscripted variable 1)
DCW NAME3     (Address of subscripted variable 2)
DCW NAME4     (Address of subscripted variable 3)
DCW LABEL     (Address of field where computed
               address is to be stored)
      or
      001     (X1)
      002     (X2)
      003     (X3)
DCW nnn       (Size of NAME1)
```

### *XAMIN*

The XAMIN macro generates the e*xamine* subroutine. The data-name to be examined, the size and class of the data-name, switches defining the various options and literal 1 (and literal 2, if existing) are specified.

The examine subroutine replaces and tallies according to the options used. The linkage used for the examine subroutine is:

```
B     XMN
DCW   NAME1    (Data-name)
DCW   000      (Not subscripted)
      001      (Subscripted)
DCW   nnn      (Length of data-name)
DCW   009      (Class is numeric)
      00X      (Class is alphanumeric)
DCW   ABC      (A: R = Replacing
                   T = Tallying
                B: 1 = All
                   2 = Leading
                   3 = Until First
                   4 = First
                C: R = Replacing
                   b = No Replacing)
DCW   LIT1     (Name of literal 1)
DCW   LIT2     (Name of literal 2, if present)
      000      (Not present)
```

## COBOL Compiler Output

The output from the COBOL compiler is on the devices specified in the ASGN cards. The LIST file output, with an assumed assignment to the printer, is composed of:

*Source Program Listing.* A listing of the COBOL source program is output during the Syntax Scan section of the COBOL compiler. The listing is made up of the card images and their sequence. An asterisk (*) in the space between the card sequence and the card image is a sequence-error flag. An example of a source-program listing is shown in the sample program in *Appendix IV.*

*Dictionary.* A COBOL dictionary is output immediately following the source-program listing. The dictionary equates source special-names to symbolic names, source data-names to symbolic names, and source procedure-names to symbolic names. An example of a dictionary is shown in the sample program in *Appendix IV.* An M in the space between TYPE and NAME indicates that the particular name is used to refer to more than one field (multidefined) and requires qualification.

*Qualification Table.* This table is output only when name qualification is stipulated in the source program. Names that require qualification and refer to only one specific field appear once in the qualification table. Names that require qualification and refer to more than one specific field appear more than once in the qualification table.

If WORK5 (an optional logical file) is assigned, the output is a listing of the Autocoder symbolic statements generated by the COBOL compiler. The total number of diagnostic messages is output at the end of the listing. An END OF COMPILATION message is also output.

### COBOL Diagnostic Messages

Incorporated within the compiler are three general types of COBOL diagnostic messages. These diagnostics are intended for use by the source programmer and are displayed on the device assigned to the LIST file.

The first type of diagnostic message is name-associated. These messages appear as flags in the dictionary that associates COBOL source names or COBOL source qualified-names with the equivalent Autocoder names used by the compiler. These flags refer to such errors as COBOL keywords used as data-names. The following example of the equating of Autocoder and COBOL names demonstrates how name-associated diagnostics appear.

```
            TYPE        NAME      SOURCE
              .           .         .
              .           .         .
              .           .         .
            SPEC      M A11       CARDS
            SPEC        A12       PRINTER
            SPEC        A13       OVERFLOW
            SPEC        A14       LAST-CARD
            FILE        A15       STATUS-FILE
             REC      M A16       CARDS
              DATA      A17       CARD-ORDER
**KEYWORD     DATA      A18       DIGIT
              DATA      A19       RECORD
              .           .         .
              .           .         .
              .           .         .
```

In the preceding example, the COBOL keyword DIGIT was used as a name, and is flagged. In addition, the name CARDS was used more than once in the source program and is flagged as being multidefined. If CARDS has been qualified in all its uses, no error exists.

The second type of diagnostic message is in the form of a listing, describing problems encountered while analyzing the sentence structure of the source program. An error in COBOL-prescribed sentence structure automatically suspends compilation after the entire source program has been analyzed. In the listing, an assigned sequence number of the card containing the statement in error is related to the item expected by the compiler in contrast to the specific item actually encountered. If, to continue analysis of the source program, the compiler drops any portion of the source program, the item(s) dropped are included as a part of the diagnostic message. In special cases, the expected portion of the diagnostic message may be re-

placed by a language term or a rule describing a requirement of the COBOL language, such as

```
ENVIRONMENT DIVISION MUST PRECEDE DATA DIVISION
```

In correcting program errors indicated by this type of diagnostic, it is recommended that the programmer analyze the diagnostics in sequential order. Further, the correcting process can be greatly facilitated if the programmer compares the source program error-statement with the corresponding COBOL-prescribed format as outlined in the SRL publication *COBOL (on Disk) Specifications for IBM 1401, 1440, and 1460,* Form C24-3235. The user is reminded that any items that are dropped as a result of an error in sentence structure are not analyzed by the compiler prior to their deletion.

An example of analyzing COBOL diagnostics is:

```
COBOL Souce Program
  SEQUENCE      CARD IMAGE
     .              .
     .              .
     .              .
    30  ENVIRONMENT DIVISION.
    40  CONFIGURATION SECTION.
    50  SOURCE-COMPUTER. IBM-1401 NO-OVERLAP.
    60  OBJECT-COMPUTER. IBM-1401 NO OVERLAP.
    70  SPECIAL-NAMES.
     .              .
     .              .
     .              .
   230  PROCEDURE DIVISION.
     .              .
     .              .
     .              .
   280  PARAGRAPH2.
   290      IF COBOLZ IS EQUAL TO REF1 OR LESS
   300        OR GREATER THAN 2 OR NOT LESS
   310        THAN 7 THEN GO TO COBOLN.
   320  PARAGRAPH3.
     .              .
     .              .
     .              .

COBOL Diagnostics
  SEQUENCE          COMMENTS
     50      '.'  =NO-OVERLAP=
    300      INVALID SUBJECT OR OBJECT  =OR=
```

First consider the diagnostic with the lowest sequence number:

```
50      '.'  =NO-OVERLAP=
```

The diagnostic means that in sequence number 50, the compiler expected a period. Instead of a period, NO-OVERLAP was sensed. Referring to the reference-format section of the COBOL specifications publication, the source computer paragraph is defined.

```
SOURCE-COMPUTER.  { IBM-1401 }
                  { IBM-1440 }
                  { IBM-1460 }
```

NO-OVERLAP is not an entry for the SOURCE-COMPUTER paragraph. (However, NO-OVERLAP is an optional entry for the OBJECT-COMPUTER paragraph.)

Consider the diagnostic:

```
300  INVALID SUBJECT OR OBJECT  =OR=
```

The diagnostic indicates that the error is in sequence number 300. The compiler expected a valid subject or a valid object; OR was sensed. Referring to the COBOL specifications publication, the format for relational conditions is defined.

```
[ [ { data-name            } ] { IS [NOT] GREATER THAN }
[ [ { literal              } ] { IS [NOT] LESS THAN    }
[ [ { arithmetic expression} ] { IS [NOT] EQUAL TO     }
                               { =                     }

                { data-name            }
                { literal              }
                { arithmetic expression}
```

In the source statement (sequence 290 and 300), IF COBOLZ IS EQUAL TO REF1 OR LESS OR GREATER THAN 2 OR NOT LESS there should be a data-name, or a literal, or an arithmetic expression after LESS for comparison with the implied subject, COBOLZ. Instead, OR was encountered by the compiler.

The meanings of the symbols used in the sentence-structure-error diagnostic message are:

| *Symbol* | *Meaning* |
|---|---|
| PROCEDURE-NAME | Blanks bound COBOL-language terms and rules |
| 'SECTION' | Quotes bound literal values expected by the compiler. |
| =RAN= | Equal signs bound literal values encountered by the compiler. |
| (ALL TO MASTER) | Parentheses bound literal values dropped by the compiler. |
| 'RUN'/LITERAL | The slash represents the word OR. |

Additional examples of the sentence-structure-error diagnostic are:

```
620 UNDECLARED NAME =TAPE-ORDER= (TAPE-ORDER)
```

620 is the assigned sequence number of the card containing the statement in error; UNDECLARED NAME is a language term, bounded by blanks, representing an error encountered by the compiler; = TAPE ORDER = is the undeclared name, bounded by equal signs; and

(TAPE-ORDER) is the item dropped, bounded by parentheses.

650 'TO' =ALL= (ALL TO MASTER)

650 is the assigned sequence number of the card containing the statement in error; 'TO' is the literal value, bounded by quotes, expected by the compiler; =ALL= is the literal value, bounded by equal signs, encountered by the compiler; and (ALL TO MASTER) is the item dropped, bounded by parentheses.

The third category of diagnostics consists of warning messages inserted as comments cards in the Autocoder instructions generated by the compiler. Messages of this category are preceded by either a single asterisk (*), or double asterisks (**).

A message preceded by a single asterisk indicates that a possible error in logic has occurred. These errors may or may not be intentional. For example, Field A is longer than Field B. The program contains an entry to move Field A to Field B. The COBOL compiler recognizes this is irregular and issues the diagnostic, I00 A-FIELD EXCEEDS B-FIELD, in the flow of Autocoder instructions. If the instruction were issued for the purpose of truncating Field A in the move (which would occur in this instance), the diagnostic would serve only as a reminder to the programmer that he had issued an unusual move instruction. However, if the programmer unknowingly issued such a move, his error would be pointed out to him by the COBOL compiler.

A message preceded by double asterisks indicates a more serious error. These errors should be corrected in order that the resulting object program operate correctly. For example, if the OPEN verb is not used in the source program, the diagnostic, A11 FILE MUST BE OPENED, is output in the flow of Autocoder instructions if a READ or WRITE instruction is used in the source program. This diagnostic can also appear when a file is declared in the DATA DIVISION and is not opened in the PROCEDURE DIVISION. This type of error would preclude the successful operation of an object program.

If further explanation of the error condition is desired, the diagnostic message identifications (in these cases I00 and A11) reference the diagnostic messages with explanatory notes shown in Figure 1.

When the third category of diagnostics occurs in a compilation, one of two messages is printed on the printer. If the diagnostics were preceded by a single asterisk, the message

TOTAL NUMBER OF DIAGNOSTICS — *nnn*

is printed on the printer.

If one or more of the diagnostics were preceded by double asterisks, the system temporarily halts, and the message

NOTE NUMBER OF ERRORS NEEDING CORRECTION — *nnn*
TOTAL NUMBER OF DIAGNOSTICS — *nnn*
PRESS START TO CONTINUE START RESET AND
START TO BYPASS JOB

is printed on the printer. The user then has the option of continuing or bypassing the job by following the directions given in the message. The user is advised that diagnostics preceded by double asterisks may cause Autocoder source program errors to be generated by the COBOL compiler.

| Reference | Division | DIAGNOSTIC and Meaning |
|---|---|---|
| A01 | Data (IOCS) | UNIT RECORD FILES MUST HAVE STANDARD RECORD SIZES<br>Card Reader 80 characters<br>Punch 80 characters<br>Printer 100 or 132 characters (1403)<br>144 characters or less (1443) |
| A02 | Data (IOCS) | UNIT RECORD FILES MUST HAVE FORM1 RECORDS<br>Blocked records are not permitted in unit record files. |
| A04 | Data (IOCS) | FORM3 RECORDS NOT PERMITTED<br>Unblocked variable length records are not permitted. |
| A05 | Environment (IOCS) | ONLY TAPE FILES MAY RESERVE ALTERNATE AREAS<br>Tape files are the only files supported by the alternate area feature. |
| A06 | Environment (IOCS) | UNIT RECORD FILES MUST NOT RESERVE ALTERNATE AREAS<br>All alternate areas needed for unit record files are reserved automatically by the processor. Alternate areas must not be reserved for unit record files. Example:<br>SELECT READER ASSIGN TO 1402-R RESERVE 1 ALTERNATE AREA |
| A07 | Data (IOCS) | FILE-LIMITS NOT SPECIFIED<br>All disk file MD statements require a FILE-LIMITS entry. Each address must be 6 characters. |
| A08 | Environment (IOCS) | INVALID DISK UNIT<br>The following are the only valid disk units:<br>1301-D or 1311-D: 0, 1, 2, 3, or 4 |
| A09 | Data (IOCS) | UNIT RECORD FILE/1301-LABEL RECORDS MUST BE OMITTED<br>The LABEL RECORDS clause must be specified as OMITTED for unit record files and/or 1301. |
| A10 | Data (IOCS) | FORM 4 RECORDS—BLOCK SIZE MUST BE IN CHARACTERS<br>The block size on Form 4 records must be designated using the key word CHARACTER. Therefore, the following format for Form 4 records is invalid:<br>BLOCK CONTAINS 400 RECORDS |
| A11 | Procedure (IOCS) | FILE MUST BE OPENED<br>The OPEN verb is used to initiate the processing of one or more input and/or output files. At least one of the two clauses must be written. The clause can specify one or more files to be opened. |
| A12 | Environment (IOCS) | FILE MUST BE SELECTED<br>Each file to be processed by the object program must be named in a SELECT file-name entry and the designated name must be unique within the source program. |
| A13 | Data (IOCS) | BLOCK SIZE MUST NOT EXCEED 4 DIGITS<br>The maximum block size that can be specified may contain four digits including leading zeros. |
| A14 | Data (IOCS) | RECORD SIZE MUST NOT EXCEED BLOCK SIZE<br>A record size specification must always be smaller than the block size specification that is to contain the record. |
| A15 | Data (IOCS) | DATA RECORD DECLARED WITHOUT SPECIFYING SIZE<br>01 following a file description not declared legally. |
| A16 | Data (IOCS) | INVALID FILE LIMITS<br>The upper file limit must be the address of the first sector of the last physical record in the file. |
| D01 | Data | PICTURE EXCEEDS 30 CHARACTERS<br>A picture may not contain more than 30 characters. Example: The picture 9(450) contains six actual characters, and therefore, will be properly declared. If the above picture were written with 450 9's, it would not be a valid picture description and the picture would be dropped by the processor. |
| D02 | Data | CONFLICTING EDITING CLAUSES<br>Zero suppress, check protect, and float dollar sign edit clauses are mutually exclusive. The last clause specified is the one chosen for use. It is suggested that the more complex editing be done using a picture clause. |
| D03 | Data | MUST NOT SPECIFY DECIMAL AT GROUP LEVEL<br>Only elementary items can use this option. |
| D04 | Data | MUST NOT SPECIFY EDITING AT GROUP LEVEL<br>Only elementary items can use this option. |

Figure 1. Diagnostic Messages (Part 1 of 4)

| Reference | Division | DIAGNOSTIC and Meaning |
|---|---|---|
| D05 | Data | CONFLICT BETWEEN PICTURE AND SIZE CLAUSE<br><br>If there is a contradiction between a size clause and a picture clause, the picture specifications take precedence over the size clause. For example, the clause "SIZE IS 10" would conflict with a picture defined as 9(6). In this case, a size of 6 would be used by the compiler for compilation. |
| D06 | Data | CONFLICT BETWEEN PICTURE AND POINT LOCATION CLAUSE<br><br>If there is a contradiction between point location and a picture clause, the picture specifications take precedence over the point location clause.<br><br>Example: PICTURE IS 9V99<br>POINT LEFT 3.<br><br>In this example, the point will precede the second character from the left, as specified by the picture. |
| D07 | Data | CONFLICT BETWEEN PICTURE AND CLASS<br><br>If there is a contradiction between class and a picture clause, the picture specifications take precedence over the class clause.<br><br>Example: PICTURE IS 999<br>CLASS IS ALPHAMERIC.<br><br>In the example, the picture is specified as numeric; therefore, the class must also be specified as numeric. |
| D08 | Data | CONFLICT BETWEEN PICTURE AND EDITING CLAUSE<br><br>If there is a contradiction between editing and a picture clause, the picture specifications take precedence over the editing clause.<br><br>Example 1: PICTURE IS 999 CHECK PROTECT.<br>(CHECK PROTECT clause is ignored.)<br><br>Example 2: PICTURE IS $$9 CHECK PROTECT.<br>(Floating dollar sign will be used.) |
| D09 | Data | ELEMENTARY ITEM MUST HAVE SIZE OR PICTURE CLAUSE<br><br>An element of data (elementary item) is a piece of data which is never further divided. Each elementary item must have either SIZE clause or PICTURE clause designation in the DATA DIVISION. |
| D11 | Data | A RECORD MUST NOT EXCEED 999 CHARACTERS<br><br>All records may be as large as 999 characters. |
| D12 | Data | RLI MUST BE 4 CHARACTERS<br><br>The Record Length Indicator (RLI) must be four characters in length including leading zeros. |
| D13 | Data | RECORD SIZES WITHIN THIS FILE MUST NOT CONFLICT<br><br>More than a single 01 level within a FD or MD is considered implied redefinition and results in a redefinition of the first 01. If the record size is variable, the DEPENDING clause should be used. This diagnostic was given because record lengths within the FD or MD were not the same and could cause an error at object time. |
| D14 | Data | INVALID EDITING<br><br>Illegal editing was specified and all editing for the entry in error was omitted by the processor. Some conditions which would cause invalid editing are: (1) invalid symbols in a picture clause; (2) size of editing field exceeds 120 characters; (3) high-order single zero suppress; (4) DB (debit symbol) or CR (credit symbol) not in right- or left-most position. |
| D99 | Data | PICTURE CLAUSE INVALID WITH GROUP ITEM<br><br>Only elementary items may be described with the PICTURE clause. |
| E03 | Environment | HARDWARE DEVICE MULTI-DEFINED<br><br>Hardware devices may not be multi-defined. The example below illustrates two combinations that will cause errors which will cause a multi-defined device.<br><br>Statements (1) and (2) or (1) and (3) would cause a multi-defined device because 1402-P is assigned to PUNCH in SPECIAL-NAMES. Statements (2) and (3) excluding statement (1) would cause a multi-defined device because two files cannot be assigned to the same hardware device.<br><br>Example: SPECIAL-NAMES.<br>(1) 1402-P, 4 IS PUNCH.<br>INPUT-OUTPUT SECTION.<br>FILE CONTROL.<br>(2) SELECT FILE2 ASSIGN TO 1402-P<br>(3) SELECT FILE3 ASSIGN TO 1402-P. |
| E51 | Procedure (IOCS) | I/O TABLE OVERFLOW<br><br>The maximum number of files is 12. Some error conditions cause extra entries to be made in the I/O table. |
| E52 | Procedure (IOCS) | WRITE REFERENCES INPUT FILE<br><br>Only OUTPUT and INPUT-OUTPUT files can be referenced by the WRITE verb. Correct the OPEN statement or the WRITE statement. |

Figure 1. Diagnostic Messages (Part 2 of 4)

| Reference | Division | DIAGNOSTIC and Meaning |
|---|---|---|
| E53 | Procedure (IOCS) | INVALID KEY USED IN WRITE ON INPUT-OUTPUT FILE<br><br>This type file requires a READ prior to each unique WRITE on the file. The INVALID KEY statement associated with the READ obviates the need for the INVALID KEY option of the WRITE statement. |
| E54 | Procedure (IOCS) | READ REFERENCES OUTPUT FILE<br><br>Only INPUT and INPUT-OUTPUT files may be referenced by the READ verb. Correct the OPEN or the READ statement. |
| F01 | Procedure | END STATEMENTS NOT ASSOCIATED WITH THE FOLLOWING FILES<br><br>There must be at least one implicit or explicit AT END statement associated with every READ statement. Once an AT END statement has been executed, an attempt to READ from the associated file will constitute an error unless a subsequent CLOSE and OPEN have been executed for that file. |
| H01 | Procedure | INVALID USE OF SUBSCRIPTING<br><br>This diagnostic occurs when invalid subscripting is used without an OCCURS clause.<br><br>Error Example: DATA DIVISION.<br>.<br>.<br>.<br>FILE SECTION.<br>01 NAME1<br>02 NAME2 (***NOTE)<br>03 NAME3<br><br>PROCEDURE DIVISION.<br>.<br>.<br>.<br>(***NOTE) MOVE NAME2 (3) TO WORKAREA<br><br>The procedure statement specified subscripting; therefore, NAME2 must contain an OCCURS clause. |
| I00 | Procedure | A-FIELD EXCEEDS B-FIELD<br><br>The field being moved is larger in size than the receiving field. The field being moved will be truncated to the length of the receiving field. |
| I01 | Procedure | NON-NUMERIC FIELD USED IN COMPUTATION<br><br>This diagnostic is given on arithmetic fields that are not defined as numeric and are referred to by IF POSITIVE or IF NEGATIVE. |
| I02 | Procedure | EDIT MASK TOO SMALL<br><br>A GIVING, COMPUTE, or MOVE using edited fields too small to accept the field being placed into the edited mask will get this diagnostic.<br><br>Example: Moving field 12345 to an edit mask $$ 9.99 is erroneous because this edit mask will accept only four characters. |
| I03 | Procedure | A-FIELD EXCEEDS 18 DIGITS<br><br>Fields used in arithmetic computations and expressions must not exceed 18 digits. |
| I05 | Procedure | B-FIELD EXCEEDS 18 DIGITS (Refer to I03) |
| I06 | Procedure | NON-NUMERIC FIGCON USED IN COMPUTATION<br><br>No figurative constants can be used in arithmetic computations. |
| I07 | Procedure | INVALID SOURCE<br><br>When meaningless source is compiled, this diagnostic will be given followed by a STOP RUN expansion. |
| I08 | Procedure | INTERMEDIATE RESULTS MUST NOT EXCEED 20 DIGITS<br><br>Fields requiring decimal alignment in arithmetic computations must not have intermediate results that expand beyond 20 digits. |
| I09 | Procedure | ALPHANUMERIC TO NUMERIC MOVE<br><br>This diagnostic will be given if a group item (declared as numeric) is moved to an elementary numeric field. All group items are classed as alphanumeric. This is a warning diagnostic and does not necessarily constitute an error. |
| I10 | Procedure | INVALID SOURCE—IN PRECEDING STATEMENT<br><br>This diagnostic is given when meaningless source is compiled in conjunction with the DISPLAY verb.<br><br>Example: DISPLAY data-name/literal ON (name) instead of the correct format<br>DISPLAY data-name/literal UPON (name) |
| I11 | Procedure | ROUNDED EXCLUDES LEFT JUSTIFY CLAUSE<br><br>When left justification is used in conjunction with the ROUNDING option, justification ignored during compilation. |

Figure 1. Diagnostic Messages (Part 3 or 4)

| Reference | Division | DIAGNOSTIC and Meaning |
|---|---|---|
| 112 | Procedure | GROUP ITEMS MUST NOT HAVE IMPLIED DECIMAL<br><br>Decimal alignment is neglected when group items classed as numeric are moved. |
| 113 | Procedure | GROUP ITEM MUST NOT BE USED IN COMPUTATION<br><br>Only elementary items can be used in arithmetic computations. |
| 115 | Procedure | INVALID USE OF GROUP ITEM<br><br>1. When the conditional IF POSITIVE or IF NEGATIVE is used, the field referred to must be elementary.<br><br>2. When the conditional IF NUMERIC or IF ALPHABETIC is used, the field referred to should be elementary. If the fields are not elementary, the diagnostic is then given as a reminder to review possible zoning in the units position of the fields being tested for these conditions. |
| 117 | Procedure | ITEM EXCEEDS 20 DIGITS<br><br>Numeric fields set up for computational purposes must not exceed 20 digits. When a numeric field exceeding 20 digits is compared, the diagnostic is given and instructions to perform a non-numeric compare are given. |
| 118 | Procedure | CLASS CONTRADICTION<br><br>This diagnostic is given as a reminder to review classes of fields being operated on which have contradicting classes. For example, a numeric field compared to a non-numeric figurative constant would cause this diagnostic to be given. |
| 119 | Procedure | INVALID USE OF EDITING<br><br>Editing must be done on data fields using procedural statement with the appropriate verbs. For example:<br><br>1. The field referred to by the ACCEPT verb must not have editing.<br><br>2. The name of the integer that designates the number of times a PERFORM statement is to be executed in a PERFORM verb TIMES statement must not have editing symbols.<br><br>3. Sending field must be numeric and greater than 1 digit. |
| 120 | Procedure | SIZE OF LITERAL MUST EQUAL 1<br><br>This literal must be one character in length; e.g., in the EXAMINE verb. |
| 121 | Procedure | INVALID SUBSCRIPTING<br><br>An example of subscripting that will cause this diagnosis is:<br><br>DATA DIVISION.<br>01 TABLE.<br>02 NAME 1 OCCURS 3 TIMES SIZE IS 2.<br>PROCEDURE DIVISION.<br>MOVE NAME 1 (1, 2) --------<br><br>Note the double subscripting indicated in the procedural statement contradicting the declaration for NAME1. |

Figure 1. Diagnostic Messages (Part 4 of 4)

## Jobs

The COBOL system performs four major operations.

1. Compiles source programs.
2. Assembles Autocoder-IOCS programs.
3. Produces object programs.
4. Starts the execution of object programs.

Because these operations are performed by the four processors of the system, the operations are called *processor jobs.* In this respect, the COBOL compiler compiles source programs, the Autocoder processor assembles Autocoder-IOCS programs, the Output processor produces object programs, and the Execution processor starts the execution of object programs.

One other operation, updating the COBOL system, is also considered a job. Updating the COBOL system is called an *update job.* Update jobs are described in *Updating a COBOL 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 are discussed in the following section *(Preparing Processing 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 COBOL system defines the files and assigns them to specific input/output devices. In the description that follows of preparing individual processor jobs, any file assignment that the user must make is explained.

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

The last card of a COBOL source-program deck is the END OF SOURCE card. The format of this card follows.

| *Columns* | *Contents* |
|---|---|
| 1-3 | END |
| 5-6 | OF |
| 8-13 | SOURCE |

The END OF SOURCE card signals the COBOL compiler that the entire source program has been read into core storage by the INPUT file. Figure 2 shows a COBOL source deck.

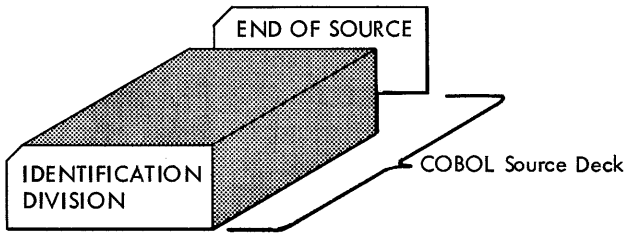



Figure 2. COBOL Source Deck

### 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.

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

```
COBOL RUN
COBOL RUN THRU AUTOCODER
COBOL RUN THRU OUTPUT
COBOL 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 devices. This entry refers to the devices 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 as a result of the job.

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.
8. Arrangement. This entry references a figure that shows the manner of arranging card input for the job.

*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.

## COBOL RUN

This is the type of run that results in Autocoder symbolic statements. When this run is performed, only the COBOL compiler is selected. To get a machine-language object program, the Autocoder statements must be processed by the Autocoder and Output processors.

This is the only type of run that can be performed when the Autocoder system and the COBOL system reside on separate disk units.

*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. COBOL diagnostic messages, if errors are sensed in the source program.
2. Source-program listing.
3. COBOL dictionary.
4. Qualification table, if name qualification is stipulated in the source program.
5. Punched-card deck containing the Autocoder symbolic statements.

*Output Option Available.* Listing of the Autocoder symbolic statements. To obtain this option, use a WORK5 ASGN card.

*Required User Assignments.* None.

*Control Cards.*

1. If a listing of the Autocoder symbolic statements is desired on the printer, punch the ASGN card in the following manner. If used, this ASGN card must precede the RUN card.

| *Columns* | *Contents* |
|---|---|
| 6-10 | WORK5 |
| 16-19 | ASGN |
| 21-29 | PRINTER *n* |

The value *n* represents the number of print positions. If the printer is a 1403, a 1 indicates 100 positions and a 2 indicates 132 positions. If the printer is a 1443, a 1 indicates 120 positions and a 2 indicates 144 positions.

2. The RUN card, the only required control card, is punched in the following manner.

| *Columns* | *Contents* |
|---|---|
| 6-10 | COBOL |
| 16-18 | RUN |

*Arrangement.* The arrangement of input cards is shown in Figure 3.

*Note:* If a cross reference list is desired instead of a label table when an AUTOCODER RUN, an AUTOCODER RUN THRU OUTPUT, or an AUTOCODER RUN THRU EXECUTION is performed, change column 31 of the compiler-generated CTL card to blank.

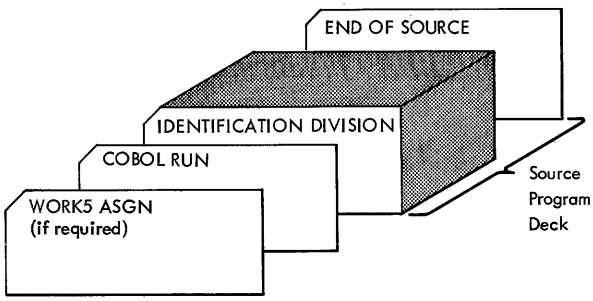



Figure 3. COBOL RUN

## COBOL RUN THRU AUTOCODER

This is the type of run that results in Autocoder text. When this run is performed, the COBOL compiler and the Autocoder processor are selected. To get a machine-language object program, the Autocoder text must be processed by the Output processor.

*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. COBOL diagnostic messages, if errors are sensed in the source program.
2. Source-program listing.
3. COBOL dictionary.
4. Qualification table, if name qualification is stipulated in the source program.
5. Autocoder diagnostic messages, if errors are sensed.

6. Label table.
7. Autocoder text and a message specifying the START address of the text.

*Output Options Available.* Listing of the Autocoder symbolic statements. To obtain this option, use a WORK5 ASGN card.

*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 | 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. If a listing of the Autocoder symbolic statements is desired on the printer, punch the ASGN card in the following manner. If used, this ASGN card must precede the RUN card.

| *Columns* | *Contents* |
|---|---|
| 6-10 | WORK5 |
| 16-19 | ASGN |
| 21-29 | PRINTER *n* |

   The value *n* represents the number of print positions. If the printer is a 1403, a 1 indicates 100 positions and a 2 indicates 132 positions. If the printer is a 1443, a 1 indicates 120 positions and a 2 indicates 144 positions.

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

| *Columns* | *Contents* |
|---|---|
| 6-10 | COBOL |
| 16-18 | RUN |
| 21-24 | THRU |
| 26-34 | AUTOCODER |

*Arrangement.* The arrangement of input cards is shown in Figure 4.

## COBOL RUN THRU OUTPUT

This is the type of run that results in compiling and assembling a machine-language object program. When this run is performed, the COBOL compiler, the Autocoder processor, and the Output processor are selected.

*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. COBOL diagnostic messages, if errors are sensed in the source program.
2. Source-program listing.
3. COBOL dictionary.
4. Qualification table, if name qualification is stipulated in the source program.
5. Autocoder diagnostics messages, if errors are sensed.
6. Label table.
7. Object-program listing.
8. Object program in the condensed-loader format (six-card loader for 1401 or 1460, seven-card loader for 1440).

*Output Options Available.* Listing of the Autocoder symbolic statements. To obtain this option, use a WORK5 ASGN card.

*Required User Assignments.* None.

*Control Cards.*

1. If a listing of the Autocoder symbolic statements is desired on the printer, punch the ASGN card in the following manner. If used, this ASGN card must precede the RUN card.

| *Columns* | *Contents* |
|---|---|
| 6-10 | WORK5 |
| 16-19 | ASGN |
| 21-29 | PRINTER *n* |

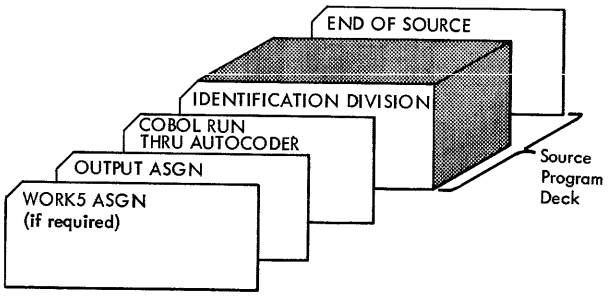



Figure 4. COBOL RUN THRU AUTOCODER

The value *n* represents the number of print positions. If the printer is a 1403, a 1 indicates 100 positions and a 2 indicates 132 positions. If the printer is a 1443, a 1 indicates 120 positions and 2 indicates 144 positions.

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

| *Columns* | *Contents* |
|---|---|
| 6-10 | COBOL |
| 16-18 | RUN |
| 21-24 | THRU |
| 26-31 | OUTPUT |

*Arrangement.* The arrangement of input cards is shown in Figure 5.

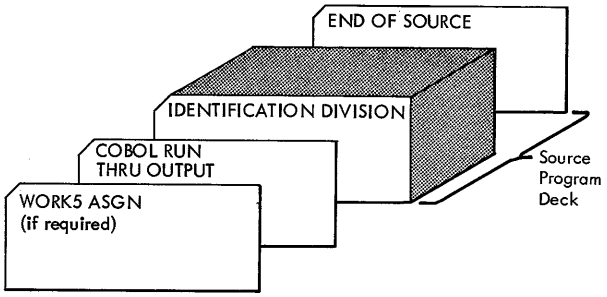



Figure 5. COBOL RUN THRU OUTPUT

## COBOL RUN THRU EXECUTION

This is the type of run that results in compiling, assembling, and executing a machine-language object program. When this run is performed, the COBOL complier, Autocoder processor, Output processor, and Execution processor are selected. This run is similar to the standard load-and-go operation.

*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.*

1. COBOL diagnostic messages, if errors are sensed in the source program.
2. Source-program listing.
3. COBOL dictionary.
4. Qualification table, if name qualification is stipulated in the source program.
5. Autocoder diagnostic messages, if errors are sensed.
6. Label table.
7. Autocoder program listing.
8. Object program in the coreload format and a message specifying the START and END addresses of the program that is stored in the CORELOAD file in disk storage.

*Output Options Available.* Listing of the Autocoder symbolic statements. To obtain this option, use a WORK5 ASGN card.

*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.*

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 | 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* indicates 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. If a listing of the Autocoder symbolic statements is desired on the printer, punch the ASGN card in the following manner. If used, this ASGN card must precede the RUN card.

| *Columns* | *Contents* |
|---|---|
| 6-10 | WORK5 |
| 16-19 | ASGN |
| 21-29 | PRINTER *n* |

The value *n* represents the number of print positions. If the printer is a 1403, a 1 indicates 100 positions and a 2 indicates 132 positions. If the printer is a 1443, a 1 indicates 120 positions and a 2 indicates 144 positions.

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

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

*Arrangement.* The arrangement of input cards is shown in Figure 6.

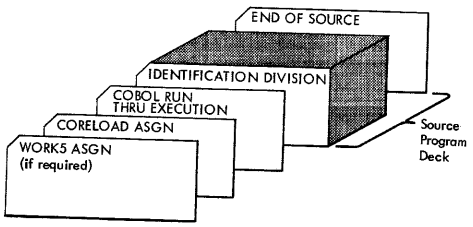



Figure 6. COBOL RUN THRU EXECUTION

| ASGN Card Format | | | Assumed Assignment | Remarks |
|---|---|---|---|---|
| Label Field (Columns 6-15) | Operand Field (Columns 16-20) | Operand Field (Columns 21-72) | | |
| SYSTEM | ASGN | {1311 UNIT n / 1301 UNIT 0} | 1311 unit -- user-assigned<br>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 COBOL 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 | If the CONTROL file and the INPUT file are assigned to the card reader, the assignment must be to the same card reader. |
| 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. If the MESSAGE file and the LIST file are assigned to the printer, the assignment must be to the same printer. |
| 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 of the Autocoder system develops a 100-character program listing. If the MESSAGE file and the LIST file are assigned to the printer, the assignment must be to the same printer. |
| INPUT | ASGN | {READER n / 1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn} | READER 1 | If the CONTROL file and the INPUT file are assigned to the card reader, the assignment must be to the same card reader. |
| OUTPUT | ASGN | {PUNCH n / 1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn / OMIT} | PUNCH 4 (1401 and 1460)<br>PUNCH 1 (1440) | |
| LIBRARY | ASGN | {1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn} | 1311 UNIT 0, START 012900, END 019980<br>1301 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. If the MESSAGE, LIST, and WORK5 files are assigned to a printer, the assignment must be to the same printer. |
| WORK1 | ASGN | {1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn} | 1311 UNIT 0, START 007200, END 009400<br>1301 UNIT 0, START 007200, END 009400 | |
| WORK2 | ASGN | {1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn} | 1311 UNIT 0, START 007200, END 009400<br>1301 UNIT 0, START 007200, END 009400 | |
| WORK3 | ASGN | {1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn} | 1311 UNIT 0, START 009400, END 012900<br>1301 UNIT 0, START 009400, END 012900 | |
| WORK4 | ASGN | {1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn / OMIT} | OMIT | |
| WORK5 | ASGN | {1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn / PRINTER n / OMIT} | OMIT | |
| CORELOAD | ASGN | {1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn / OMIT} | OMIT | |

Figure 7. ASGN Card Formats and Assumed Assignments

## Changing File Assignments

Each logical file defined by the COBOL 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. The uses of the logical files should be considered when deciding file assignments.

### 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 7. Valid device entries are shown in Figure 8.

Leave a blank between items in the operand field as shown in Figure 7. If, for example, the OUTPUT file is to be assigned to disk area 004000 through 004799 on 1311 unit 1, the user would code the ASGN card for punching as shown in Figure 9. The END address to

| Device Entry and Values of n and nnnnnn | Remarks |
|---|---|
| {1311 / 1301} UNIT n, START nnnnnn, END nnnnnn<br><br>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.<br><br>The values of nnnnnn must adhere to the following rules:<br>1. WORK1 and WORK2 files. 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.<br>2. WORK3, WORK4, and WORK5 files. The START and END addresses (1311 and 1301) must be multiples of 10.<br>3. LIBRARY file. The START and END addresses (1311 and 1301) must be multiples of 20.<br><br>If these rules are violated, the system automatically narrows in the disk area to an area that does adhere to these rules. |
| READER n<br><br>For 1402, n can be 0, 1, or 2.<br><br>For 1442, n can be 1 or 2. | For 1402, n represents the pocket into which the cards are stacked.<br>For 1442 and 1444, n represents the number of the unit. |
| PUNCH n<br><br>For 1402, n can be 0, 4, or 8.<br><br>For 1442, n can be 1 or 2.<br><br>For 1444, n must be 3. | |
| PRINTER n<br><br>n can be 1 or 2 | n represents the number of print positions available on the 1403 or 1443.<br><br>For 1403, a 1 indicates 100 positions and a 2 indicates 132 positions.<br>For 1443, a 1 indicates 120 positions and a 2 indicates 144* positions.<br><br>*Only 132 print positions are used by the COBOL system. |
| CONSOLE PRINTER | The console printer must be an IBM 1447 without a buffer feature. |
| OMIT | Select this option when the file is not to be used by the COBOL system. LIST, OUTPUT, WORK4, WORK5, and CORELOAD are the only files that can be omitted. |

Figure 8. Valid Device Entries

| Label | Operation | OPERAND |
|---|---|---|
| OUTPUT | ASGN | 1311, UNIT, 1, , START, 004000, , END, 004800 |

Figure 9. Coding for OUTPUT ASGN Card

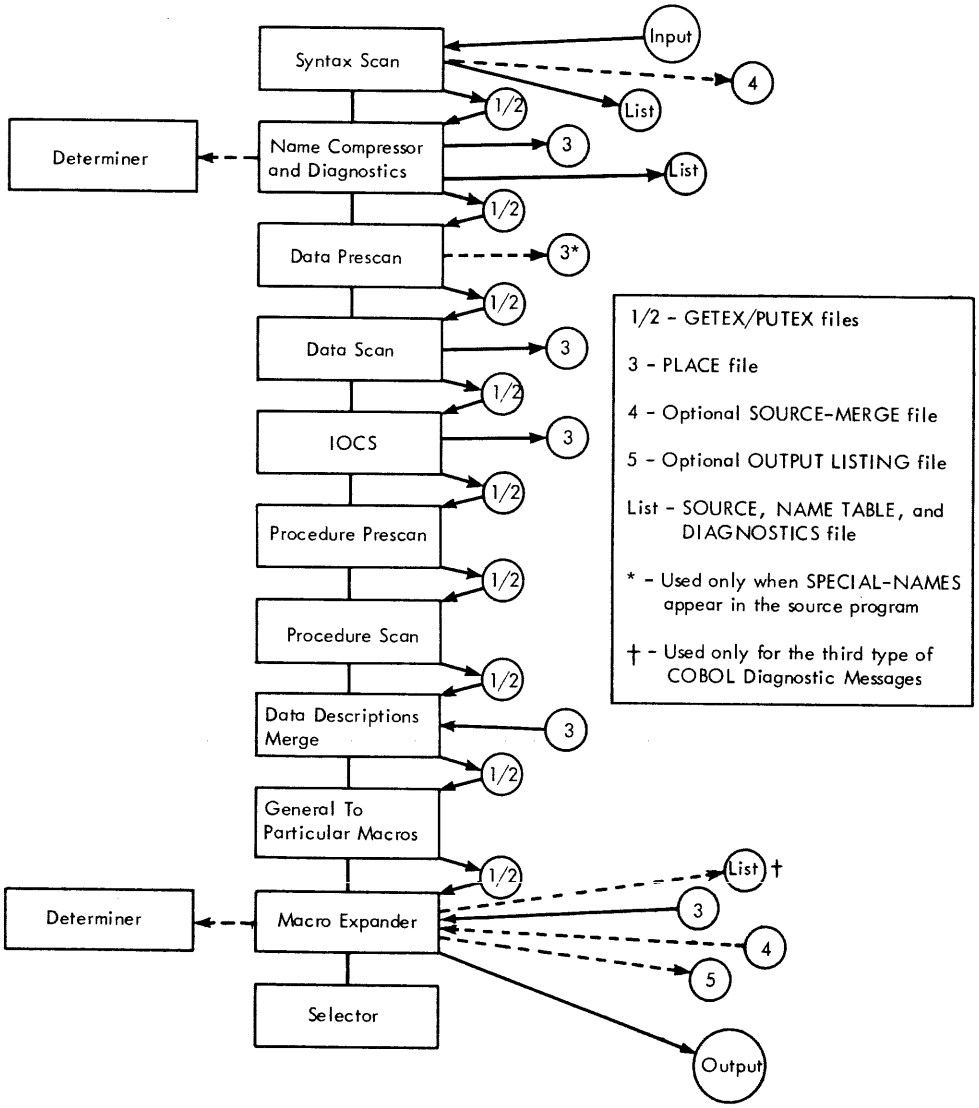



Figure 10. COBOL Compiler and Use of Logical Files

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.

*SYSTEM File.* If the SYSTEM file resides on 1311, drive 0 should be on-line because the System Control Program's assumed assignments are on drive 0. If drive 0 is not on-line, the user must use ASGN cards to change the assumed assignments for the LIBRARY, WORK1, WORK2, and WORK3 files.

*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.

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

*Note:* Do not assign the OUTPUT file to a disk area for COBOL RUN THRU OUTPUT

*WORK1 File and WORK2 File.* WORK1 and WORK2 are required files. They can be assigned to any available area in disk storage. The WORK1 and WORK2 files are used by the COBOL compiler for the GETEX and PUTEX functions that perform the large volume of data handling during compilation.

*WORK3 File.* WORK3 is a required file. It can be assigned to any available area in disk storage. The WORK3 file is used by the COBOL compiler as an out-of-line file (PLACE) that bypasses data around major portions of the compiler. In addition, if the THRU option is specified in the RUN card, WORK3 acts as an interface file between COBOL and Autocoder.

*WORK4 File.* WORK4 is an optional file. When WORK4 is assigned to any available area in disk storage, it is used by the COBOL compiler as the SOURCE-MERGE file. This function intersperses COBOL source statements, by paragraph, into the Autocoder symbolic statements generated by the COBOL compiler.

*WORK5 File.* WORK5 is an optional file. When WORK5 is assigned to any available area in disk storage or to the printer, it is used by the COBOL compiler as the COBOL OUTPUT LISTING file. This file is a listing of the Autocoder symbolic statements generated by the COBOL compiler.

### Timing Considerations

Each segment of the COBOL compiler uses specified logical files that include: INPUT, OUTPUT, LIST, GETEX/PUTEX (WORK1/WORK2), PLACE (WORK3), the optional SOURCE-MERGE (WORK4), and the optional OUTPUT LISTING (WORK5). Figure 10 is a block diagram showing the logical segments and the order in which they operate. In addition, the logical files used by the individual segments are shown.

Manipulation of the COBOL files enables the user to achieve the best possible results in the operation of the COBOL system. To attain these desired results, it must be remembered that seek time is the most significant factor affecting input/output operations time in the system. Therefore, it would be expedient for the user with a multi-unit system to distribute the COBOL files to all of the units, thus making a significant reduction in seek time.

WORK1 and WORK2 must be given further consideration if optimum seek time is to be realized. Because these two files handle large amounts of data, inefficient use of the files results in an increase of COBOL time requirements. To minimize this effect on the one-unit user, the assumed assignments for WORK1 and WORK2 are such that the two WORK files are assigned to the same area of the disk unit, as shown in Figure 11. The System Control Program "splits" each cylinder, causing WORK1 to occupy the upper half of each cylinder and WORK2 to occupy the lower half of each cylinder. A programmed false cylinder-overflow is forced as each half cylinder is operated upon and the next upper or lower cylinder is used.

The user is advised that when ASGN cards are used to change logical file assignments, the assignments affect not only the logical files used by COBOL, but also the logical files used by Autocoder. Autocoder requires that the WORK1 and WORK2 files be assigned to separate disk areas. Therefore, when the assumed assignments

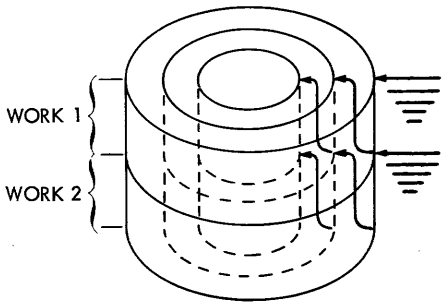



Figure 11. WORK1 and WORK2 Assigned to Same Disk Area

for WORK1 and WORK2 are changed by using ASGN cards, these files must be assigned to separate areas on disk.

#### 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.

### *Batched Files*

In some cases it would be expedient for the user to exercise the batching capability of the COBOL system. The batched-files concept applies to the external files INPUT, OUTPUT, and LIST. (The CORELOAD file, used by the Output processor of the Autocoder portion of the system, can also be batched.) The contents of these files represent one or more sequential sets of input to or output from the processors.

An example of the batching principle is as follows. Three source programs are stored sequentially on the INPUT file. The user-specified output is to be in the form of Autocoder-IOCS symbolic statements. This implies that only the COBOL compiler is to be accessed.

An INPUT ASGN card containing the START address of the beginning of the first source program and the END address of the end of the third source program is required. An OUTPUT ASGN card is required, if the OUTPUT file is to be assigned to disk. Following the OUTPUT ASGN card are three COBOL RUN cards (one RUN card for each of the three source programs on disk).

At the completion of processing of each source program, the first disk address used by the next program to be written on the OUTPUT file is printed on the MESSAGE file. Thus, the user would know not only the START and END addresses of the entire OUTPUT file, but also the START and END addresses of each of the partially processed programs.

## 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.

*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 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), 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 System Control Program from the CONTROL file.

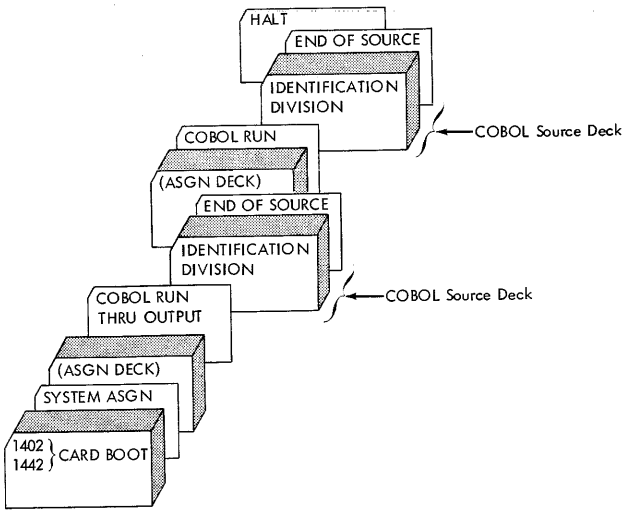



Figure 12. Stack with CONTROL and INPUT Files Assigned to the Same Device

Figure 12 shows a stack with CONTROL and INPUT files assigned to the same device.

Figure 13 shows a stack with CONTROL and INPUT files assigned to different devices.

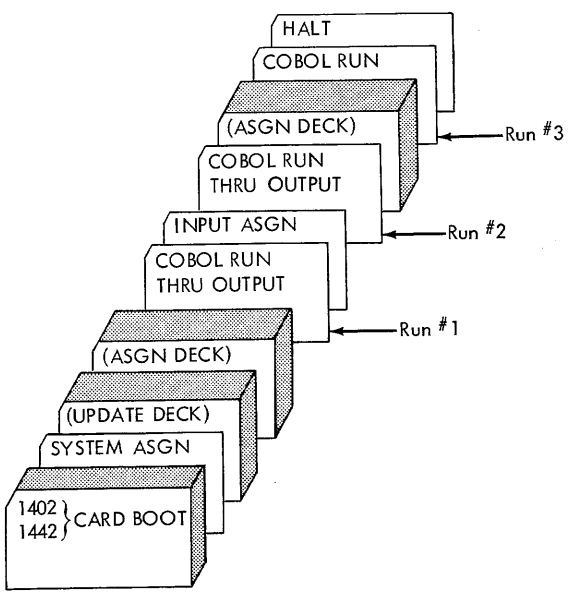
### *Running a Stack*

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.



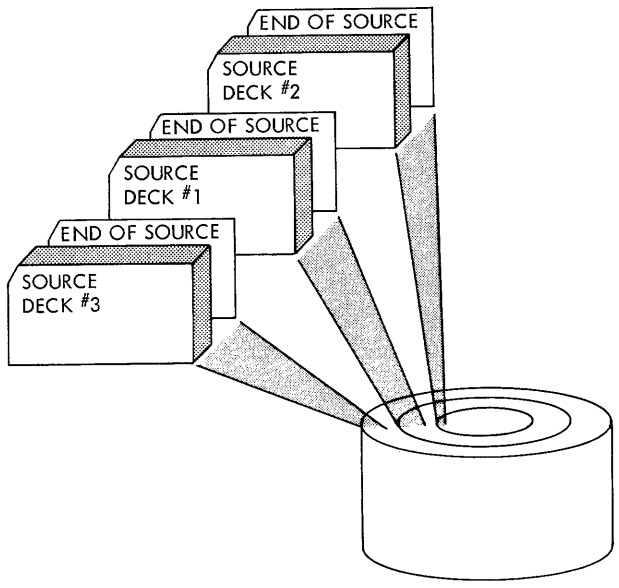




Figure 13. Stack with CONTROL and INPUT Files Assigned to Different Devices

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 COBOL 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 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. Nonprocess run out the cards in the card reader.
2. Place the last three cards (two nonprocessed 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. Nonprocess 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. Nonprocess 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.

*Note:* For a description of the preceding formats, see the Systems Reference Library publication *Autocoder (on Disk) Program Specifications and Operating Procedures for IBM 1401, 1440, and 1460,* Form C24-3259.

### Halts and Messages

The halts and messages shown in Figure 14 can appear during a stack run. To display halt numbers, press the A-address register key. Messages are printed on the MESSAGE file.

Conditions may arise that the system recognizes as being instrumental in causing a failure. In these instances, the system automatically calls in a storage- and file-print program. The contents of core storage and the WORK files are printed on the MESSAGE file, and the system continues by accepting a new job. Among the conditions that may cause a system failure are:

1. An end-of-file indication was sensed by the compiler before the end of the file was reached.
2. An excessive number of disk read or disk write errors occurred on the WORK files.
3. An error in COBOL-prescribed sentence structure occurred when coding the source program. This type of error is detected in the phase of the processor that analyzes the particular part of the program. For example, an error in subscripting would be detected in phase H01.

In certain types of halts, the operator can call in the storage- and file-print program by a manual branch to address 900. These types of halts are in the form *xxxx*, where *x* is numeric.

If the Autocoder preprocessor recognizes conditions that make it impossible to complete an operation, a hard halt occurs. In such cases, the linkage to the System Control Program is destroyed. This type of halt is in the form *xx*bb, where *x* is numeric. When the preprocessor has program control, the storage- and file-print program cannot be used. If a failure occurs, the system does not call in the program, and the operator cannot call it by a manual branch to address 900. The operator can restart by using the Card Boot, followed by the necessary job decks.

| Halt Number (A-Address Register) | MESSAGE and/or Meaning | Restart Procedure |
|---|---|---|
| 1bb | Card read error. | 1. 1402 card reader: nonprocess run out the cards in the reader. Place the last three cards (two nonprocessed cards and the card in error) in the hopper. Press START.<br>2. 1442 card reader: nonprocess run out the cards in the reader. Place the two nonprocessed cards in the hopper. (The first nonprocessed 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 disk-write operation. | Press START for ten disk-read or disk-write retries. |
| 3bb | Parity error sensed ten times during a disk-read or disk-write operation. | Press START for ten disk-read or disk-write retries. |
| 4bb | Not-ready condition sensed when a disk-read or disk-write operation was attempted. | Ready the disk unit and press START. |
| 5bb | 1. Librarian-control OPTN card is incorrect, or<br>2. Preprocessor phase not on the SYSTEM file. | 1. Nonprocess run out the cards in the card reader, correct the OPTN card, and restart the system, or<br>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:<br><br>ERROR HEADER ABOVE UNKNOWN<br><br>A phase-update card specifies a phase name that is not in the phase table.<br><br>ERROR NO KNOWN TYPE OF UPDAT<br><br>Columns 21- ? of a phase-update card are incorrect.<br><br>ERROR CYLINDER OVERFLOW<br><br>The phase-update card specifies that the phase is to be placed on a set of sectors that exceeds one cylinder.<br><br>ERROR ACTUAL IDENT UNEQUAL TO HEADER IDENT<br><br>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.<br><br>ERROR NON CONTROL CARD WITHOUT CONTROL PRECEDING<br><br>An update card is missing, out of sequence, or mis-punched.<br><br>ERROR UNKNOWN EXECUTE CARD<br><br>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.<br><br>ERROR PATCH ABOVE OUTSIDE OF PROGRAM LIMITS<br><br>The phase area cannot contain the data specified in the change cards. | The contents of the error cards are printed. Nonprocess 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 14. Halts and Messages (Part 1 of 5)

| Halt Number (A-Address Register) | MESSAGE and/or Meaning | Restart Procedure |
|---|---|---|
| | ERROR CHARACTER COUNT TOO LARGE<br><br>A change card contains a character count greater than 67 characters. The character count is punched in columns 4 and 5.<br><br>ERROR ABOVE CARD CREATES GROUP MARK WORD MARK<br><br>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 can neither be read from nor written in disk storage. | |
| 7bb | More than 30 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<br><br>This halt indicates one of the following conditions:<br>1. An INSER, DELET, or END card is missing or mispunched.<br>2. An attempt to insert or delete entries in a library routine that does not exist.<br>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:<br>1. 1402 card reader: nonprocess run out the cards in the reader. Place the END card in the hopper. Press START.<br>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.<br>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. |
| 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:<br>1. 1402 card reader: nonprocess run out the cards in the reader. Place the END card in the hopper. Press START.<br>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.<br>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 14. Halts and Messages (Part 2 of 5)

| Halt Number (A-Address Register) | MESSAGE and/or Meaning | Restart Procedure |
|---|---|---|
| 001 | WRONG SYSTEM<br><br>The message appears unconditionally on the printer. | 1. Nonprocess run out the cards in the reader.<br>2. Correct the SYSTEM ASGN card, or place the correct pack on the unit indicated in the SYSTEM ASGN card.<br>3. Restart the stack. |
| 002 | TEN RD TRIES PRESS STRT FOR 10 MORE<br><br>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<br><br>The SYSTEM ASGN card did not immediately follow the Card Boot. | 1. Nonprocess run out the cards in the reader.<br>2. Place the SYSTEM ASGN card and the remainder of the stack in the read hopper.<br>3. If the reader is 1402, press START.<br>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 times during disk bootstrap operation. | Press START for 10 disk-read retries. |
| 005 | End-of-file sensed in SYSTEM file during disk bootstrap operation. | Nonprocess run out the cards in the reader and restart the stack. |
| 006 | HALT card image<br><br>Indicates the end of the stack. | Hard halt. |
| 007 | Card-punch error. | 1. 1402 card punch: nonprocess run out the cards in the punch. Discard the last three cards (two nonprocessed cards and the card in error) in the stacker. Press START.<br>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: nonprocess run out the cards in the reader. Place the last three cards (two nonprocessed cards and the card in error) in the hopper. Press START.<br>2. 1442 card reader: nonprocess run out the cards in the reader. Place the two nonprocessed cards in the hopper. Press START on the reader and START on the console. |
| 009 | Printer error. | 1. 1403 printer: press START.<br>2. 1443 printer: press START on the printer and START on the console. |
| 010 | Nonblank card at the punch station in the 1442 card read-punch. | Nonprocess run out the cards in the 1442. Place blank cards before the nonprocessed 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<br><br>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. Nonprocess run out the cards in the reader. Place the corrected ASGN card and the two nonprocessed cards in the hopper. Press START.<br>2. 1442 card reader: nonprocess run out the cards in the reader. The first nonprocessed card is the incorrect ASGN card. Correct the ASGN card. Place the corrected ASGN card and the second nonprocessed card in the hopper. Press START on the reader and START on the console. |

Figure 14. 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 10successive 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. |
| 2930 | NOTE NUMBER OF ERRORS NEEDING CORRECTION - nnn TOTAL NUMBER OF DIAGNOSTICS - nnn PRESS START TO CONTINUE START-RESET AND START TO BYPASS JOB | 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<br>All cards not recognized by the System Control Program are flagged (***), written on the MESSAGE file, and bypassed by the System. | |
| | Card image<br><br>INVALID UPDAT TYPE<br><br>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<br><br>All header cards must have 24232 in columns 1 through 5. | |
| | Card image<br><br>PHASE AREA EXCEEDED | |
| | ****PROCESSOR UNKNOWN**** | |

Figure 14. Halts and Messages (Part 4 of 5)

| Halt Number (A-Address Register) | MESSAGE and/or Meaning | Restart Procedure |
|---|---|---|
| | CORELOAD NOT ASSIGNED, OPTION NOT DONE<br><br>The next output option is processed. | |
| | CORELOAD FILE NOT ASSIGNED, OPTION NOT DONE AND EXECUTION SUPPRESSED | |
| | Image of an output option card - OPTION UNKNOWN<br><br>The next output option card is processed. | |
| | CORELOAD HEADER — (52 positions), ID — (5 positions)<br><br>Use the information in an EXECUTION RUN card. | |
| | CORELOAD OUTPUT COMPLETE ON {1311 / 1301} UNIT n, START nnnnnn, END nnnnnn<br><br>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<br><br>ADDRESS nnnnnn | |
| | XXXXX MACRO NOT IN LIBRARY<br><br>The macro requested (XXXXX) is not in the LIBRARY file. | |
| | END OF LISTING OPTN<br><br>The library-listing job has been completed. | |
| | XXXXX BLOCKS LEFT EOJ<br><br>The library-change operation has been completed. XXXXX is the number of blocks available in the LIBRARY file. | |
| | END OF SYSTEM OPTN<br><br>The update operation has been successfully completed. | |
| | LIBRARY FILE NOT RECOGNIZED<br><br>The library has not been assigned correctly or the library has not been initialized. | |
| | OUTPUT FILE NOT ASSIGNED TO DISK<br>The RUN card specifies AUTOCODER RUN or COBOL RUN THRU AUTOCODER. The Autocoder text must be written on disk. | |
| | INPUT FILE NOT ASSIGNED TO DISK<br><br>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<br><br>1EOFb sensed in the INPUT file and the Autocoder Text has not been processed; or, the assigned INPUT file does not contain text. | |

Figure 14. Halts and Messages (Part 5 of 5)

### *COBOL-System Deck Description and Preparation*

The program card deck supplied to the user contains six sections as shown in Figure 15. One section, Marking Program, is used to separate the sections for ease in labeling the various components of the complete deck. One section, System Control Modification, is used to modify the System Control Program. Three sections, Write File-Protected Addresses, COBOL Update, and COBOL Macros, are used to build the system. The sixth section, Sample Program, is used to test the system built by the user. The individual sections are separated by marking program control cards. In the instances where there is more than one set of cards making up a section, a marking program control card separates the sets.

All cards in the system deck, except for the two 1402 load-card sets, the two 1442 load-card sets, the COBOL Macros, and the 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

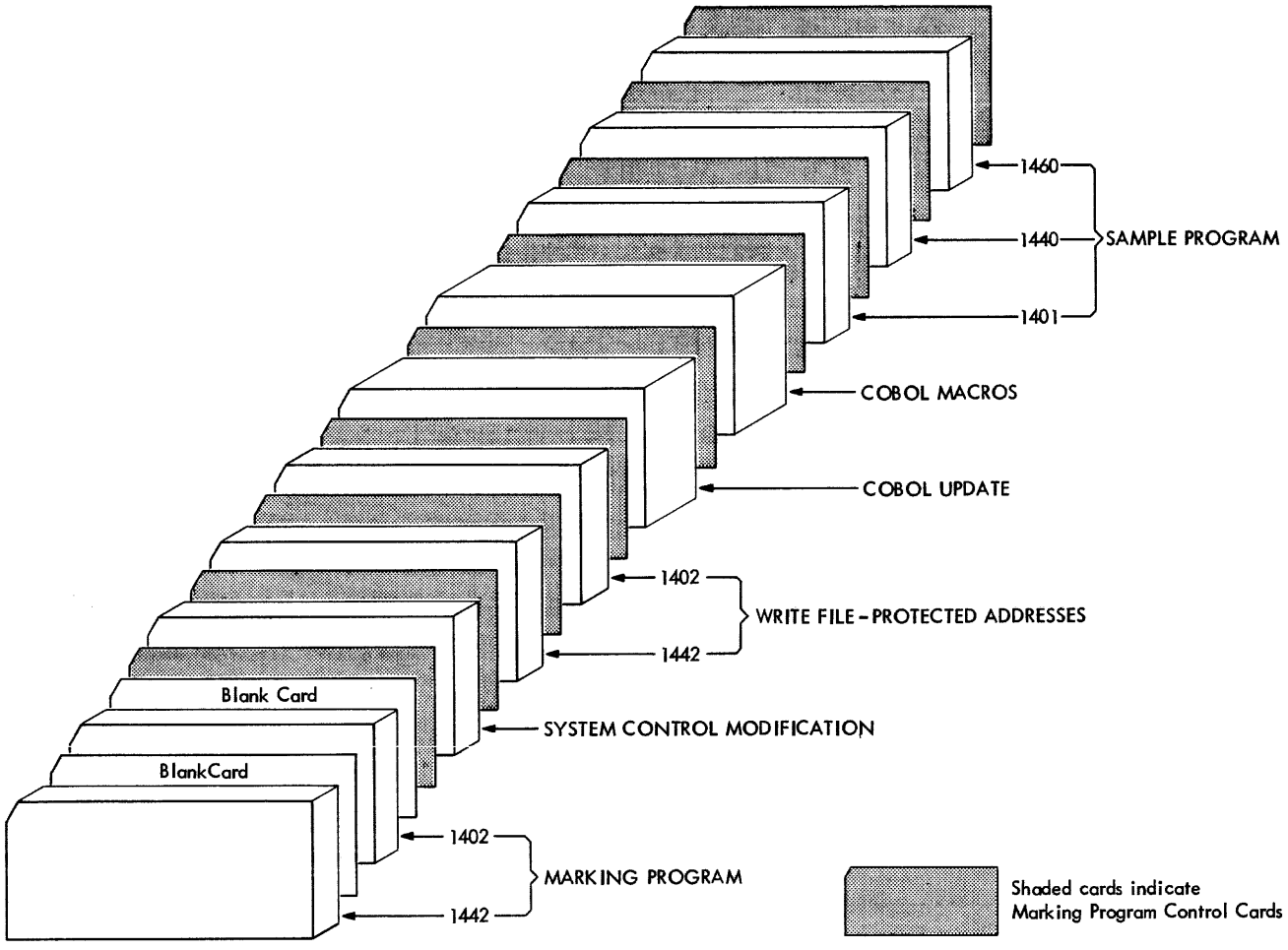



Figure 15. COBOL Program Deck

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.

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

1. Sort on columns 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 two 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 two sets of 1442 load cards.
7. Sort the remainder of the system deck on columns 75, 74, 73, and 72. After sorting, the COBOL 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 COBOL Macros and Sample Programs on column 5. After sorting, the COBOL Macros will be in the reject pocket.
10. Sort the Sample Programs on columns 4, 3, 2, 1, and 80.
11. Check the program listing and insert the Sample Programs.
12. Sort the COBOL Macros on columns 4, 3, 2, 1, 80, 79, 78, 77, and 76.
13. Check the program listing and insert the COBOL Macros.

### 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 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 COBOL 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.
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:* Nonprocess run-out the cards. Remove the last three cards in the stacker and place them in the hopper. Press START. *For the 1442:* Nonprocess run-out the cards. Place the two nonprocessed 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. |

### System Control Modification

The system Control Modification deck is punched in the Autocoder condensed-loader format and the UPDAT control-card format. The deck is made up of approximately 5 cards. The UPDAT cards are identified by the code 50CB1 punched in columns 76-80. The modification cards are identified by the code 50S*x*1 punched in columns 76-80, where *x* is alphameric. The function of the deck is to modify the System Control Program and the assumed logical file assignments of the System Control Program.

### 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.

### COBOL Update

The COBOL Update deck is punched in the Autocoder condensed-loader format and the UPDAT control-card format. The deck is made up of approximately 3300 cards, and contains the phases of the COBOL compiler. The UPDAT cards are identified by the code 50CB1 punched in columns 76-80; the COBOL phases are identified by the code 50*xx*1 punched in columns 76-80, where *x* is alphameric. The function of the deck is to load the COBOL compiler phases on the disk unit, thus permitting a COBOL run.

### COBOL Macros

The COBOL-macros deck is punched in the Autocoder library card format. The deck contains approximately 1800 cards and is identified by the code 50M*x*1 punched in columns 76-80, where *x* is alphameric. This deck places the COBOL object-time subroutines and macro instructions that set switches during assembly on the Autocoder Macro Library. COBOL requires that these macros be present during COBOL-output assembly.

### COBOL Sample Program

The COBOL Sample Program consists of approximately 250 cards. The 1401 deck is identified by the code SAMPLE-1 punched in columns 73-80. The 1440 deck is identified by the code SAMPLE-2 punched in columns 73-80. The 1460 deck is identified by the code SAMPLE-3 punched in columns 73-80. This source deck, written in the COBOL language, is used to test the effectiveness of the system built by the user.

## *Building a COBOL 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 COBOL system.

Figure 16 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 007199, and in the move mode from 007200 to 019979. The Clear Disk Storage Program applicable to the user's system can be used for this operation.

Figure 16.1 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 |

| *Columns* | *Contents* |
|---|---|
| 1-15 | L00030000719900 |
| 21-35 | M00720001997900 |

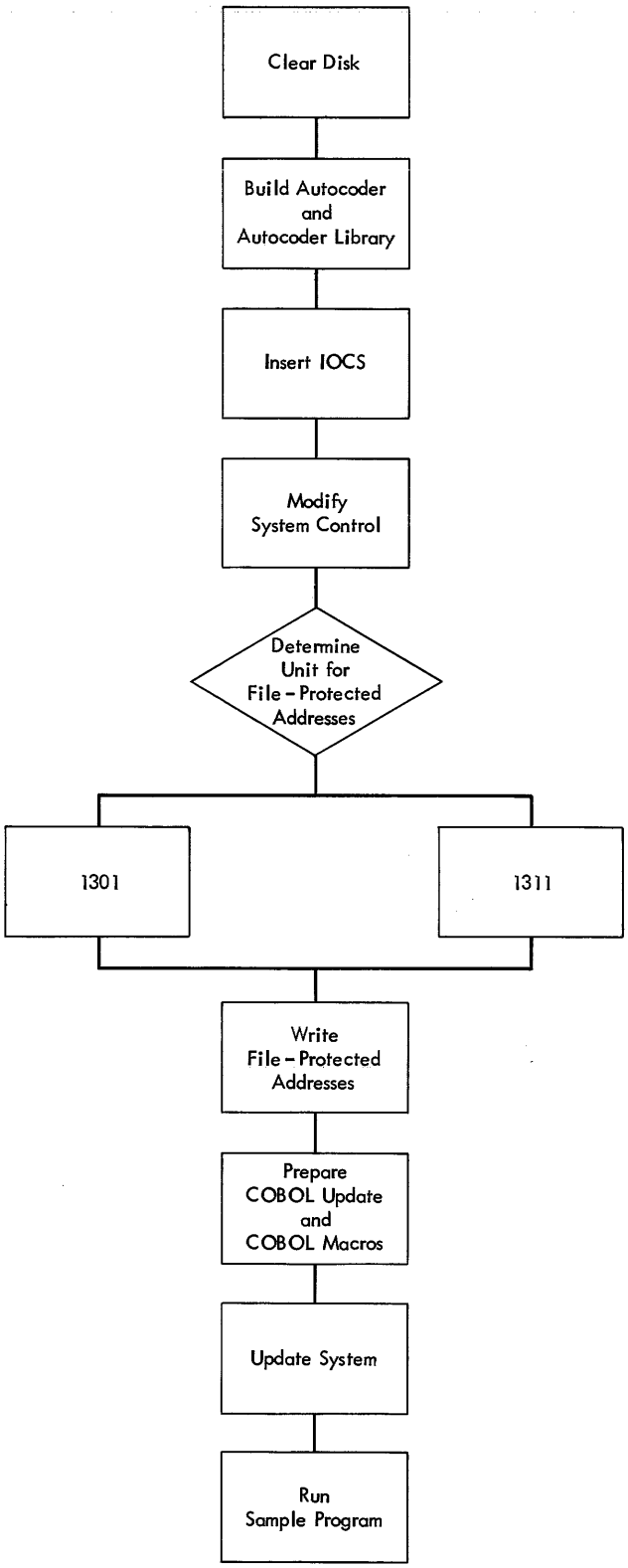



Figure 16. Building the COBOL System.

For 1301,

| *Columns* | *Contents* |
|---|---|
| 1-15 | M000000000199‡‡ |
| 21-35 | L000200000259‡‡ |
| 41-55 | M000260000299‡‡ |

| *Columns* | *Contents* |
|---|---|
| 1-15 | L000300007199‡‡ |
| 21-35 | M007200019979‡‡ |

The user must build an Autocoder system, including the Autocoder Library. After the Autocoder system has been built, the IOCS macros must be inserted into the LIBRARY file of the Autocoder system. The procedures for building an Autocoder system and inserting the IOCS macros into the Autocoder Library are described in *Autocoder (on Disk) Program Specifications and Operating Procedures for IBM 1401, 1440, and 1460*, Form C24-3259. After the Autocoder system has been built, the user is ready to build the COBOL system.

### System Control Modification

The System Control Program must be modified. Use the deck labeled SYSTEM CONTROL MODIFICATION to perform this function. Input for the process is as follows.

1. The 1402 or 1442 Card Boot deck (supplied as a part of the Autocoder System Program deck), followed by
2. The SYSTEM ASGN card, which must be punched by the user, followed by
3. The System Control Modification deck, followed by
4. The HALT card, which is the last card of the System Control Modification deck.

To modify the System Control Program when the system 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. Place the System Control Modification deck 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.

| File | Mode | File-Protected | Sector Range |
|---|---|---|---|
| SYSTEM File | | | |
| Autocoder Preprocessor Work Area | Move | No | 000000-000089 |
| Autocoder Preprocessor | Move | No | 000090-000199 |
| Autocoder Preprocessor | Load | No | 000200-000259 |
| Autocoder Preprocessor | Move | No | 000260-000299 |
| Autocoder Preprocessor | Load | No | 000300-000899 |
| Not Used | Load | No | 000900-002499 |
| System Control Program | Load | Yes | 002500-003175 |
| Autocoder Assembler Program | Load | Yes | 003176-004799 |
| COBOL Compiler Program | Load | Yes | 004800-007199 |
| WORK1 and WORK2 Files | Move | No | 007200-009399 |
| WORK3 File | Move | No | 009400-012899 |
| LIBRARY File | Move | No | 012900-019979 |

Figure 16.1. Disk Storage Allocation

To modify the System Control Program when the system 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. Place the System Control Modification deck 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 halts that can occur when using the System Control Modification deck are shown in Figure 14.

### 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 | 004800 (prepunched) |
| 51-52 | TO (preunched) |
| 54-59 | 007200 (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 004800. 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 007200. 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. |

### COBOL Update and COBOL Macros

To build the COBOL system, the decks labeled COBOL UPDATE and COBOL MACROS are used. Input for this building process is as follows.

1. The 1402 or 1442 Card Boot deck (supplied as a part of the Autocoder system program deck), followed by
2. The SYSTEM ASGN card, which must be punched by the user, followed by
3. The COBOL UPDATE deck, followed by
4. The COBOL MACROS deck, followed by
5. 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 write-address mode switch off.
3. Set the I/O check-stop switch off.
4. Set the check-stop switch and disk-write switch on.
5. Set the mode switch to RUN.
6. Press CHECK RESET and START RESET.
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 build the system when it is to reside on 1301:

1. Set the I/O check-stop switch off.
2. Set the write-address mode 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.

The halts that can occur when using the COBOL UPDATE and COBOL MACROS decks are shown in Figure 14.

### 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 IV*. Figure 17 shows the Sample Program deck.

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 000100, END 000199 (prepunched) |

The Sample Program is prepared for a stack run in the following manner.

1. The Card Boot deck, which is supplied as part of the Autocoder system program deck, followed by
2. The SYSTEM ASGN card, which must be punched by the user, followed by
3. The CORELOAD ASGN card, followed by
4. The COBOL RUN THRU EXECUTION card, followed by
5. The source program statements and the END OF SOURCE card, followed by
6. The HALT card, which must be punched by the user.

The procedures for running the Sample Program are described in *Running a Stack.*

## *Updating a COBOL System*

The COBOL 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*.

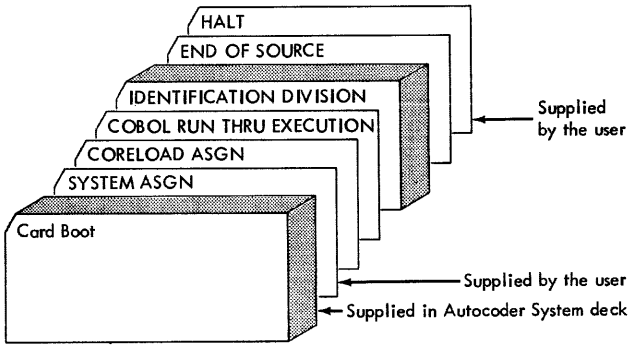



Figure 17. Sample Program

# 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 18 shows the formats of ASGN cards and the assumed assignments for the logical files. Figure 19 shows the valid device entries for the ASGN cards.

Figure 20 shows the formats of the following control cards:

Halt (HALT) card

Note (NOTE) card

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) | Operand Field (Columns 16-20) | Operand Field (Columns 21-72) | | |
| SYSTEM | ASGN | {1311 UNIT n / 1301 UNIT 0} | 1311 unit -- user-assigned<br>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 COBOL 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 | If the CONTROL file and the INPUT file are assigned to the card reader, the assignment must be to the same card reader. |
| 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. If the MESSAGE file and the LIST file are assigned to the printer, the assignment must be to the same printer. |
| 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 of the Autocoder system develops a 100-character program listing. If the MESSAGE file and the LIST file are assigned to the printer, the assignment must be to the same printer. |
| INPUT | ASGN | {READER n / 1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn} | READER 1 | If the CONTROL file and the INPUT file are assigned to the card reader, the assignment must be to the same card reader. |
| OUTPUT | ASGN | {PUNCH n / 1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn / OMIT} | PUNCH 4 (1401 and 1460)<br>PUNCH 1 (1440) | |
| LIBRARY | ASGN | {1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn} | 1311 UNIT 0, START 012900, END 019980<br>1301 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. If the MESSAGE, LIST, and WORK5 files are assigned to a printer, the assignment must be to the same printer. |
| WORK1 | ASGN | {1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn} | 1311 UNIT 0, START 007200, END 009400<br>1301 UNIT 0, START 007200, END 009400 | |
| WORK2 | ASGN | {1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn} | 1311 UNIT 0, START 007200, END 009400<br>1301 UNIT 0, START 007200, END 009400 | |
| WORK3 | ASGN | {1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn} | 1311 UNIT 0, START 009400, END 012900<br>1301 UNIT 0, START 009400, END 012900 | |
| WORK4 | ASGN | {1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn / OMIT} | OMIT | |
| WORK5 | ASGN | {1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn / PRINTER n / OMIT} | OMIT | |
| CORELOAD | ASGN | {1311 UNIT n, START nnnnnn, END nnnnnn / 1301 UNIT n, START nnnnnn, END nnnnnn / OMIT} | OMIT | |

Figure 18. ASGN Card Formats and Assumed Assignments

| Device Entry and Values of n and nnnnnn | Remarks |
|---|---|
| {1311 / 1301} UNIT n, START nnnnnn, END nnnnnn<br><br>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.<br><br>The values of nnnnnn must adhere to the following rules:<br>1. WORK1 and WORK2 files. 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.<br>2. WORK3, WORK4, and WORK5 files. The START and END addresses (1311 and 1301) must be multiples of 10.<br>3. LIBRARY file. The START and END addresses (1311 and 1301) must be multiples of 20.<br><br>If these rules are violated, the system automatically narrows in the disk area to an area that does adhere to these rules. |
| READER n<br><br>For 1402, n can be 0, 1, or 2.<br><br>For 1442, n can be 1 or 2. | For 1402, n represents the pocket into which the cards are stacked.<br>For 1442 and 1444, n represents the number of the unit. |
| PUNCH n<br><br>For 1402, n can be 0, 4, or 8.<br><br>For 1442, n can be 1 or 2.<br><br>For 1444, n must be 3. | |
| PRINTER n<br><br>n can be 1 or 2 | n represents the number of print positions available on the 1403 or 1443.<br><br>For 1403, a 1 indicates 100 positions and a 2 indicates 132 positions.<br>For 1443, a 1 indicates 120 positions and a 2 indicates 144* positions.<br><br>*Only 132 print positions are used by the COBOL system. |
| CONSOLE PRINTER | The console printer must be an IBM 1447 without a buffer feature. |
| OMIT | Select this option when the file is not to be used by the COBOL system. LIST, OUTPUT, WORK4, WORK5, and CORELOAD are the only files that can be omitted. |

Figure 19. 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 |
| Note | | NOTE | Any message and/or instruction |
| Pause | | PAUSE | Any message and/or instruction |
| Run | COBOL | RUN | |
| | COBOL | RUN | THRU AUTOCODER |
| | COBOL | RUN | THRU OUTPUT |
| | COBOL | RUN | THRU EXECUTION |

Figure 20. Control-Card Formats

# Appendix II

The name, identification, and function of each phase in the COBOL 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)<br>50X01 (1402) | Builds System Control on a disk unit. |
| Card Boot | 50SZ1 (1442)<br>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.<br>2. Initializes switches according to the type of reader, punch, and printer (serial or parallel).<br>3. Reads in the I/O package.<br>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.<br>2. Determines whether the user has exceeded specified file limits.<br>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 | |

## COBOL Compiler

This section describes the phases that make up the COBOL compiler.

| *Name* | *ID* | *Function* |
|---|---|---|
| CBL | 50CB1 | Initialization of deblocking routines (GETEX and PUTEX) and I/O buffers. |
| A01 | 50A01 | Initialization for the rest of the A and B phases. |
| A11 | 50A11 | 1. Reads and lists the source program (from the INPUT file to the LIST file).<br>2. Delimits items in the source program. |
| A02 | 50A21 | 1. Outputs Autocoder statements from source programs to PLACE file (WORK3).<br>2. Outputs source statements to SOURCE-MERGE file (WORK4). |
| A03 | 50A31 | 1. Inserts internal operators based upon pivoted key words present in tables A23 through A53.<br>2. Conditions punctuation. |
| A23 | 50A41 | Table of pivotal key words. |
| A33 | 50A51 | Table of pivotal key words. |
| A43 | 50A61 | Table of pivotal key words. |

| *Name* | *ID* | *Function* |
|---|---|---|
| A53 | 50A71 | Table of pivotal key words. |
| A14 | 50A81 | 1. Loads core storage with interpretive strings for syntax analysis.<br>2. Positions the GETEX file (WORK1/WORK2), by division, for phase A24. |
| A24 | 50A91 | 1. Analyzes the source program through the use of the interpretive strings.<br>2. Inserts diagnostic operators into the stream of source program symbols. |
| AS2 | 50201 | Interpretative strings for IDENTIFICATION and ENVIRONMENT DIVISIONS (4K systems only). |
| AT2 | 50211 | |
| AU2 | 50221 | |
| AV2 | 50231 | |
| AW2 | 50241 | |
| AX2 | 50251 | |
| AY2 | 50261 | |
| AZ2 | 50271 | |
| A-2 | 50281 | |
| AJ2 | 50291 | |
| AK2 | 502A1 | |
| AL2 | 502B1 | |
| AM2 | 502C1 | |
| AN2 | 502D1 | |
| AO2 | 502E1 | |
| AP2 | 502F1 | |
| AQ2 | 502G1 | |
| AR2 | 502H1 | |
| AA2 | 50AC1 | Interpretative strings for IDENTIFICATION and ENVIRONMENT DIVISIONS (8K, 12K, and 16K systems only). |
| AB2 | 50AD1 | |
| AS3 | 50301 | Interpretative strings for FILE SECTION of the DATA DIVISION (4K systems only). |
| AT3 | 50311 | |
| AU3 | 50321 | |
| AV3 | 50331 | |
| AW3 | 50341 | |
| AX3 | 50351 | |
| AY3 | 50361 | |
| AZ3 | 50371 | |
| A-3 | 50381 | |
| AJ3 | 50391 | |
| AK3 | 503A1 | |
| AL3 | 503B1 | |
| AM3 | 503C1 | |
| AA3 | 50AE1 | Interpretative strings for FILE SECTION of the DATA DIVISION (8K, 12K, and 16K systems only). |
| AS4 | 50401 | Interpretative strings for WORKING-STORAGE and CONSTANT SECTIONS of the DATA DIVISION (4K systems only). |
| AT4 | 50411 | |
| AU4 | 50421 | |
| AV4 | 50431 | |
| AW4 | 50441 | |
| AX4 | 50451 | |
| AY4 | 50461 | |

| *Name* | *ID* | *Function* |
|---|---|---|
| AA4 | 50AF1 | Interpretative strings for WORKING-STORAGE and CONSTANT SECTIONS of the DATA DIVISION (8K, 12K, and 16K systems only). |
| AS5 | 50501 | Interpretative strings for the PROCEDURE DIVISION (4K systems only). |
| AT5 | 50511 | |
| AU5 | 50521 | |
| AV5 | 50531 | |
| AW5 | 50541 | |
| AX5 | 50551 | |
| AY5 | 50561 | |
| AZ5 | 50571 | |
| A-5 | 50581 | |
| AJ5 | 50591 | |
| AK5 | 505A1 | |
| AL5 | 505B1 | |
| AM5 | 505C1 | |
| AN5 | 505D1 | |
| AO5 | 505E1 | |
| AP5 | 505F1 | |
| AQ5 | 505G1 | |
| AA5 | 50AG1 | Interpretative strings for the PROCEDURE DIVISION (8K, 12K, and 16K systems only). |
| A34 | 50AA1 | Supervises iterations through strings and divisions. |
| A44 | 50AB1 | Positions GETEX file (WORK1/WORK2) in relation to unprocessed data for iterations |
| A05 | 50AH1 | 1. Further source-program analysis.<br>2. Conditions source program for the rest of the compiler. |
| B01 | 50B01 | 1. Substitutes compressed names for source names and literals.<br>2. Builds name table in core storage.<br>3. Determines error conditions associated with name-qualification and multi-definitions. |
| B02 | 50B11 | Determines error conditions associated with COBOL reserved words used as names within the source program. |
| B12 | 50B21 | Table of COBOL reserved words. |
| B03 | 50B31 | Outputs COBOL dictionary on LIST file. |
| B04 | 50B41 | 1. Creates data description for literals used in the source program.<br>2. Outputs data descriptions for all items on the PLACE file (WORK3). |
| B05 | 50B51 | Outputs diagnostic messages on LIST file. Diagnostic message tables. |
| BX1 | 50B61 | |
| BX2 | 50B71 | |
| BX3 | 50B81 | |
| BX4 | 50B91 | |
| BX5 | 50BA1 | |
| BX6 | 50BB1 | |
| BX7 | 50BC1 | |
| BX8 | 50BD1 | |
| BX9 | 50BE1 | |
| BX0 | 50BF1 | |

| Name | ID | Function |
|---|---|---|
| B06 | 50BG1 | Assigns unique compressed names to multi-defined names. |
| B07 | 50BH1 | 1. Substitutes unique names when qualified.<br>2. Outputs qualified name dictionary on the LIST file. |
| C01 | 50C01 | Entire phase is executed only if a SPECIAL-NAMES paragraph appears in the source program.<br>1. Outputs special-names data descriptions on the PLACE file (WORK3).<br>2. Generates Autocoder-COBOL equate cards.<br>3. Conditions switch-names for further processing. |
| C02 | 50C11 | Optional phase. Processes level-88 items, converting appearances of condition names in the PROCEDURE DIVISION to conditional expressions. |
| C03 | 50C21 | Optional phase. Inserts move instructions into the overlay portion of the PROCEDURE DIVISION to initialize values of data items. |
| D01 | 50D01 | 1. Analyzes picture clauses to determine size, class, decimal count, and editing.<br>2. Builds a table of all 01-level names which encompass an OCCURS clause or redefinition. |
| D02 | 50D11 | 1. Builds an internal description for each entry used based upon the clauses used in the source declarations.<br>2. Updates these descriptions with information obtained from the picture analysis in Phase D01.<br>3. Generates diagnostics where discrepancies are recognized. |
| D03 | 50D21 | 1. Phase D03 is called only if editing is used.<br>2. Builds and declares edit masks and completes the edit section of the data description. |
| D13 | 50D31 | 1. Outputs editing masks on PLACE file (WORK3).<br>2. Inserts file-names into WRITE statements.<br>3. Converts WRITE-FROM statements into MOVE and WRITE statements. |
| D04 | 50D41 | 1. Builds group size by totaling the elementary sizes.<br>2. Provides padding for OCCURS clauses.<br>3. Controls nested redefinition sizes and origin flags.<br>4. Processes subscript levels. |
| D05 | 50D51 | 1. Controls word-mark placement of each entry.<br>2. Controls nested redefinition origin names.<br>3. Processes file-record equates and sizes.<br>4. Outputs storage-declaration macros. |
| D06 | 50D61 | 1. Places the data description of all entries used on the PLACE file (WORK3).<br>2. Expands across and outputs storage declarations on the PLACE file (WORK3). |
| E01 | 50E01 | Generates the JOB card. |
| E02 | 50E11 | Collects ENVIRONMENT DIVISION information in core storage for DIOCS entries. |
| E03 | 50E21 | 1. Collects DATA DIVISION information in core storage for DTF entries.<br>2. Outputs initial DTF information on the PUTEX file (WORK1/WORK2). |
| E04 | 50E31 | Processes OPEN statements and collects file-type information in core storage for DTF entries. |
| E4A | 50E41 | Optional phase. Processes the DECLARATIVES SECTION, collecting USE information in core storage for phase E05. |
| E05 | 50E51 | Processes CLOSE, READ, SEEK, and WRITE statements.<br>1. Converts disk WRITE statements to special READ statements to facilitate handling of implied INVALID KEY option.<br>2. Inserts appropriate file-limits tests and PERFORM statements for disk file READ, SEEK, and WRITE statements.<br>3. Collects rewind options for tape files from the CLOSE statements into core storage for DTF entries. |
| E06 | 50E61 | 1. Analyzes information in core storage, determining additional DIOCS entries.<br>2. Determines error conditions associated with DIOCS. |
| E07 | 50E71 | Outputs:<br>1. CTL card.<br>2. ORG card, if IOCS not used.<br>3. DIOCS entries, if IOCS is used on the PLACE file (WORK3). |
| E08 | 50E81 | Outputs additional DIOCS entries. |
| E8A | 50E91 | Outputs DIOCS associated diagnostics. |
| E09 | 50EA1 | Loads core storage with subroutines to be used by the rest of the E-phases. |
| (E10, E11 E1A, E12, and E13 operate iteratively by file.) | | |
| E10 | 50EB1 | 1. Determines additional DTF entries.<br>2. Determines I/O area requirements.<br>3. Determines errors associated with DTF entries. |
| E11 | 50EC1 | Outputs DTF entries on the PLACE file (WORK3). |
| E1A | 50ED1 | Optional phase.<br>Outputs additional DTF entries on the PLACE file (WORK3) when tape files are used in the source program. |
| E12 | 50EE1 | Outputs DTF associated diagnostics on the PLACE file (WORK3). |
| E13 | 50EF1 | Optional phase. Outputs additional DTF entries on the PLACE file (WORK3) when standard labels are used in the source program. |
| E14 | 50EG1 | Outputs:<br>1. DA statements.<br>2. MACOP macro.<br>3. LDNGO macro (if THRU EXECUTION is specified on the COBOL RUN card) on the PLACE file (WORK3). |
| F01 | 50F01 | 1. Delimits conditional expressions and the elements within them.<br>2. Conditions COMPUTE statements.<br>3. Conditions usage of subscripting. |
| F02 | 50F11 | Processes conditional statements, converting expressions to the simple relational form. |
| F03 | 50F21 | Processes READ statements. |
| F04 | 50F31 | 1. Phase F04 is called if the PERFORM verb is used.<br>2. Generates labels associated with PERFORM statements and inserts them into the PERFORM statements.<br>3. Associates "last" name of the PERFORM statement with the label and builds a table of these elements. |

| *Name* | *ID* | *Function* |
|---|---|---|
| F05 | 50F41 | 1. Searches for procedure-names and inserts "pending" labels as return-linkages.<br>2. The table built in F04 is searched for the procedure-names found and the labels are made pending. |
| G01 | 50G01 | 1. Breaks down the following COBOL source statements into macro form:<br>a. ADD<br>b. SUBTRACT<br>c. MULTIPLY<br>d. DIVIDE<br>e. MOVE<br>f. PERFORM (options 1 and 2)<br>g. READ (no AT END).<br>2. Conditions PERFORM (options 3, 4, and 5) and AT END and INVALID KEY options for further processing. |
| G02 | 50G11 | 1. Processes arithmetic operators in arithmetic expressions and COMPUTE statements.<br>2. Processes relational operators in IF and UNTIL expressions.<br>3. Passes generated labels and logic connectors to phase G03. |
| G03 | 50G21 | 1. Processes conditional statements.<br>2. Ties in generated labels for READ linkage, UNTIL, and IF with conditional statements. |
| G04 | 50G31 | 1. Expands fixed form COBOL statements to macro statements.<br>2. Creates linkage macros for PERFORM (options 3, 4 and 5).<br>3. Determines number of generated temporary buckets needed in object-run arithmetic computations. |
| G05 | 50G41 | Eliminates redundant linkages in relational statement to fully optimize these statements. |
| G06 | 50G51 | Eliminates redundant generated temporary buckets used in intermediate computations to fully optimize arithmetic statements. |
| H01 | 50H01 | 1. Collects data, device, switch, and literal descriptions and converts these descriptions to table entries.<br>2. Processes SUBSCRIPT macros.<br>3. Inserts the appropriate data-name after each occurrence of a subscript name. |
| H02 | 50H11 | 1. Sets up storage for Phases H03 and H04 one-time or iterative processing.<br>2. Table initialization.<br>3. Clears storage.<br>4. Calls H03. |
| H03 | 50H21 | 1. Builds a table of data descriptions in storage from PLACE file (WORK3).<br>2. Calls H04 when the PROCEDURE DIVISION is recognized. |
| H04 | 50H31 | 1. Merges data descriptions after every name in the PROCEDURE DIVISION that has a description.<br>2. Iterates back to H02 when necessary. |
| I01 | 50I01 | 1. Selects the appropriate lists for fixed expansions, ADVANCING option, STOP RUN, and switches.<br>2. Conditions relational macros for further processing.<br>3. Sets up parameters for EXAMINE subroutine. |
| I02 | 50I11 | 1. Sets up parameters for DISPLAY, ACCEPT, and GO TO DEPENDING subroutines.<br>2. Expands IF NUMERIC and IF ALPHABETIC macros.<br>3. Selects appropriate library lists for STOP literal.<br>4. Diagnostic scan of conditions.<br>5. Selects appropriate library instructions for READ, WRITE, OPEN, and CLOSE.<br>6. Conditions subscript macros and sets up parameters for SUBSCRIPT subroutines. |
| I03 | 50I21 | 1. Conditions input data for further processing and scans data for validity.<br>2. Conditions GIVING, POSITIVE, NEGATIVE, and relational macros for further processing.<br>3. Diagnostic scan of arithmetic expressions. |
| I04 | 50I31 | 1. Selects appropriate library lists and subroutines for MOVE and MOVE ALL macros.<br>2. Conditions POSITIVE and NEGATIVE macros for further processing. |
| I05 | 50I41 | Conditions arithmetic macros for further processing by I06.<br>a. Keeps a record of intermediate accumulators.<br>b. Calculates decimal alignment.<br>c. Optimizes library codes to be selected. |
| I06 | 50I51 | 1. Selects appropriate library lists for arithmetic and ON SIZE ERROR macros.<br>2. Conditions relational macros for further processing. |
| I07 | 50I61 | 1. Selects appropriate library lists for all arithmetic macros not processed up to this phase.<br>2. Selects appropriate library lists for the POSITIVE and NEGATIVE macros.<br>3. Performs a diagnostic scan of GIVING and GIVING (ROUNDED) macros.<br>4. Processes rounding, editing, and decimal alignment of GIVING macros.<br>5. Selects appropriate library lists for GIVING macros. |
| I08 | 50I71 | 1. Determines the type of compares to be set up for relationals.<br>2. Calculates decimal alignment where necessary for these comparisons.<br>3. Selects appropriate library lists and subroutines.<br>4. Performs a diagnostic scan of relational macros. |
| J01 | 50J01 | 1. Substitutes all parameters into model statements (JA2 through JA8) selected from library.<br>2. Provides for iteration if the tables of Phase J02 through J08 (library) will not fit in storage at one time. |
| J02 | 50J11 | Library of model statements for 4K and 8K systems only. |
| J03 | 50J21 | |
| J04 | 50J31 | |
| J05 | 50J41 | |
| J06 | 50J51 | |
| J07 | 50J61 | |
| J08 | 50J71 | |

| *Name* | *ID* | *Function* |
|---|---|---|
| JA2 | 50J81 | Library of model statements for 12K and 16K systems only. |
| JA3 | 50J91 | |
| JA4 | 50JA1 | |
| JA5 | 50JB1 | |
| JA6 | 50JC1 | |
| JA7 | 50JD1 | |
| JA8 | 50JE1 | |
| J09 | 50JF1 | 1. Sets up subroutine linkages.<br>2. Calculates, if necessary, subscript addresses.<br>3. Generates calls for COBOL macros. |
| J10 | 50JG1 | 1. Selects from PLACE file (WORK3) in the following sequence and outputs on PUTEX file (WORK1/WORK2):<br>a. JOB card<br>b. Control Card<br>c. Origin Card<br>d. DIOCS<br>e. DTF<br>f. Procedure literals<br>g. Edit Masks<br>h. Storage declarations<br>i. File areas<br>j. Generate constants, index registers, temporary storage counters, and tally register<br>k. Data literals and data moves<br>l. Calls procedure instructions from GETEX file (WORK1/WORK2). |
| J11 | 50JH1 | 1. Outputs diagnostics occurring because of conflictions between DATA DIVISION and PROCEDURE DIVISION.<br>2. Merges in Autocoder symbolic statements with diagnostics.<br>3. Calls in Phases JT1, JT2, and JT3. |
| JT1 | 50JI1 | IOCS Data and Procedure diagnostic tables. |
| JT2 | 50JJ1 | IOCS Data and Procedure diagnostic tables. |
| JT3 | 50JK1 | IOCS Data and Procedure diagnostic tables. |
| J12 | 50JL1 | 1. Optional merge of COBOL source statements into Autocoder symbolic output (from WORK4 to WORK1/WORK2).<br>2. Lists diagnostic messages (LIST file).<br>3. Converts diagnostic messages to internal note format to be inserted into Autocoder symbolic output. |
| J13 | 50JM1 | 1. Outputs Autocoder symbolic program (OUTPUT file).<br>2. Optional listing of Autocoder symbolic program (WORK5). |

| Macro Name | Subroutine Name | Subroutine Mnemonic | Approximate Size* | | Reason Macro Called |
|---|---|---|---|---|---|
| | | | M/D | No M/D | |
| ACEPT | Accept | ZAX | 349, (358)† | 349, (358)† | Use of ACCEPT verb. |
| ALCOM | Alpha Compare | YAQ | 483 | 483 | Alphabetic record with subfields compared to any data item. |
| DIVDE | Divide | DIV | 0 | 368 | Use of DIVIDE, exponentiation, or /. |
| DIVMC | | | 7 | 10 | In-line expansion of divide function. |
| DSPLY | Display | ZDY | 586, (621)† | 586, (621)† | Use of DISPLAY verb. |
| EDIT1 | Editing | ZET | 348 | 348 | Use of the following:<br>(1) COBOL zeros;<br>(2) Floating plus or minus;<br>(3) DB or single plus. |
| EXPIN | Exponentiation 1 | ZFZ | 480 | 486 | When an expression is raised by an integer exponent. |
| EXPNI | Exponentiation 2 | ZXZ | 1899 | 1930 | When an expression is raised to an exponent other than an integer. |
| FGCOM | Compare Figcon | YCL | 414 | 414 | Record with subfields being compared to a figurative constant whose SIZE is greater than 1. |
| GOTOD | Go To Depending | ZGP | 156 | 156 | Use of GO TO DEPENDING statement. |
| IFALP | If Alphabetic | YIP | 166 | 166 | All fields whose sizes are greater than 1 which are being tested for alphabetic data. |
| IFNUM | If Numeric | YIN | 148 | 148 | All fields whose sizes are greater than 1 which are being tested for numeric data. |
| INDIX | Index | ZSP | 79 | 79 | Used when any of the other subroutines are called for except multiply and divide. |
| LDNGO | | | 0 | 0 | Sets permanent switch to influence macro expansions. |
| MACOP | | | 0 | 0 | Sets permanent switch to influence macro expansions. |
| MPYMC | | | 7 | 10 | In-line expansion of multiply function. |
| MULTY | Multiply | MPY | 0 | 371 | Use of MULTIPLY, exponentiation or *. |
| MVALL | Move All | ZML | 312 | 312 | When the receiving field is a group item. |
| MVFTR | Move Field to Record | ZMR | 443 | 443 | When records of unequal length and subfields are involved in a MOVE statement. |
| OVLAY | | | 4 | 4 | Use of the VALUE clause for non-88 level data items. |
| SPLIT | Stop Literal | SLT | 133, (28)▲ | 133, (28)▲ | Use of the literal option of the STOP verb. |
| SUBS1 | Subscript 1 | XXJ | 221 | 221 | Use of single-depth subscripting. |
| SUBS2 | Subscript 2 | XXK | 269 | 269 | Use of double-depth subscripting. |
| SUBS3 | Subscript 3 | XXL | 318 | 318 | Use of triple-depth subscripting. |
| XAMIN | Examine | XMN | 356 | 356 | Use of EXAMINE verb. |

* M/D The size of the subroutine when the multiply/divide special feature is incorporated in the 1401, 1440, or 1460.

No M/D The size of the subroutine when the multiply/divide special feature is not incorporated in the 1401, 1440, or 1460.

† The first entry applies to the 1401 and 1460. The entry within the parentheses applies to the 1440.

▲ The first entry applies to a 1440 or 1460 system with a console printer. The entry within the parentheses applies to a 1440 or 1460 system with no console printer. The SPLIT macro is not applicable to a 1401 system.

Figure 21. COBOL Macros

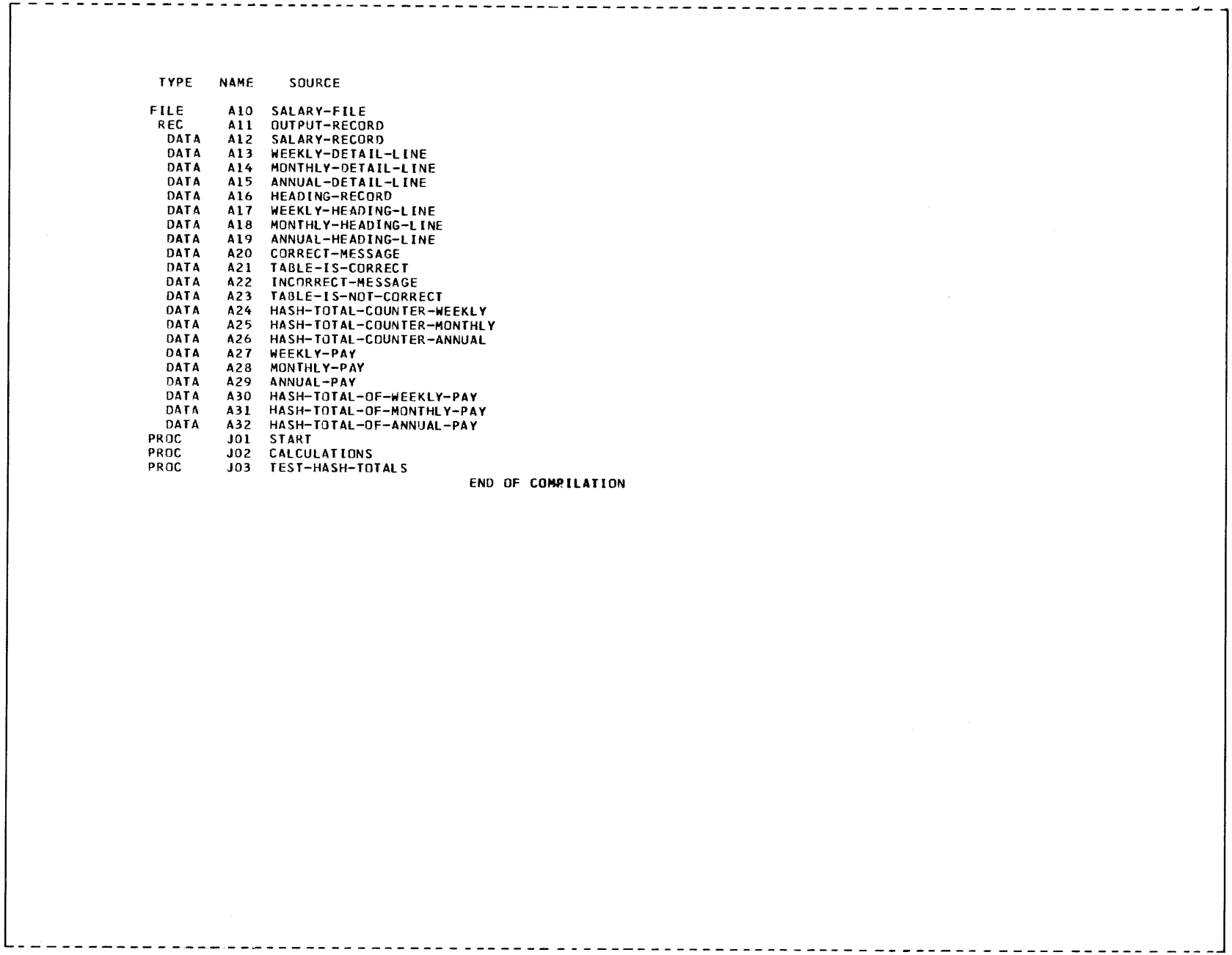
```
                                COBOL COMPILATION

SEQUENCE            CARD IMAGE                                                  IDENTIFICATION
  10   001010 IDENTIFICATION DIVISION.                                          SAMPLE-3
  20   001020 PROGRAM-ID. @COBOL SAMPLE@.                                       SAMPLE-3
  30   001030 REMARKS. A PROGRAM TO CALCULATE THE WEEKLY AND ANNUAL SALARY      SAMPLE-3
  40   001040     ASSOCIATED WITH A GIVEN MONTHLY SALARY. MONTHLY SALARY       SAMPLE-3
  50   001050     STARTS AT 500 AND IS INCREASED BY 10 UNTIL IT EQUALS 1000.   SAMPLE-3
  60   001060 ENVIRONMENT DIVISION.                                             SAMPLE-3
  70   001070 CONFIGURATION SECTION.                                            SAMPLE-3
  80   001080 SOURCE-COMPUTER. IBM-1460.                                        SAMPLE-3
  90   001090 OBJECT-COMPUTER. IBM-1460                                         SAMPLE-3
 100   001100     MEMORY SIZE 4000 CHARACTERS NO-OVERLAP.                       SAMPLE-3
 110   001110 INPUT-OUTPUT SECTION.                                             SAMPLE-3
 120   001120 FILE-CONTROL.                                                     SAMPLE-3
 130   001130     SELECT SALARY-FILE ASSIGN TO 1403-P                           SAMPLE-3
 140   001140     RESERVE NO ALTERNATE AREA.                                    SAMPLE-3
 150   002010 DATA DIVISION.                                                    SAMPLE-3
 160   002020 FILE SECTION.                                                     SAMPLE-3
 170   002030 FD  SALARY-FILE                                                   SAMPLE-3
 180   002040     LABEL RECORDS ARE OMITTED                                     SAMPLE-3
 190   002050     DATA RECORD IS OUTPUT-RECORD.                                 SAMPLE-3
 200   002060 01  OUTPUT-RECORD                PICTURE X%132¤.                 SAMPLE-3
 210   002070 WORKING-STORAGE SECTION.                                          SAMPLE-3
 220   002080 01  SALARY-RECORD.                                               SAMPLE-3
 230   002090     02  FILLER                   PICTURE X%50¤ VALUE IS SPACES.  SAMPLE-3
 240   002100     02  WEEKLY-DETAIL-LINE       PICTURE ZZZ.ZZ.                 SAMPLE-3
 250   002110     02  FILLER                   PICTURE X%5¤  VALUE IS SPACES.  SAMPLE-3
 260   002120     02  MONTHLY-DETAIL-LINE      PICTURE ZZZZ.ZZ.                SAMPLE-3
 270   002130     02  FILLER                   PICTURE X%5¤  VALUE IS SPACES.  SAMPLE-3
 280   002140     02  ANNUAL-DETAIL-LINE       PICTURE Z%5¤.ZZ.                SAMPLE-3
 290   002150 01  HEADING-RECORD.                                              SAMPLE-3
 300   002160     02  FILLER                   PICTURE X%50¤ VALUE IS SPACES.  SAMPLE-3
 310   002170     02  WEEKLY-HEADING-LINE      PICTURE A%6¤  VALUE @WEEKLY@.   SAMPLE-3
 320   002180     02  FILLER                   PICTURE X%5¤  VALUE IS SPACES.  SAMPLE-3
 330   002190     02  MONTHLY-HEADING-LINE     PICTURE A%7¤  VALUE @MONTHLY@.  SAMPLE-3
 340   002200     02  FILLER                   PICTURE X%6¤  VALUE IS SPACES.  SAMPLE-3
 350   002210     02  ANNUAL-HEADING-LINE      PICTURE A%6¤   VALUE @ANNUAL@.  SAMPLE-3
 360   003010 01  CORRECT-MESSAGE.                                             SAMPLE-3
 370   003020     02  FILLER                   PICTURE X%5¤  VALUE IS SPACES.  SAMPLE-3
 380   003030     02  TABLE-IS-CORRECT         PICTURE A%32¤ VALUE IS          SAMPLE-3
 390   003040         @    TABLE VALUES ARE CORRECT    @.                     SAMPLE-3
 400   003050 01  INCORRECT-MESSAGE.                                           SAMPLE-3
 410   003060     02  FILLER                   PICTURE X%52¤ VALUE IS SPACES.  SAMPLE-3
 420   003070     02  TABLE-IS-NOT-CORRECT     PICTURE A%28¤   VALUE IS        SAMPLE-3
 430   003080         @TABLE VALUES ARE NOT CORRECT@.                          SAMPLE-3
 440   003090 77  HASH-TOTAL-COUNTER-WEEKLY    PICTURE 9%6¤V99 VALUE IS ZERO.  SAMPLE-3
 450   003100 77  HASH-TOTAL-COUNTER-MONTHLY   PICTURE 9%6¤V99 VALUE IS ZERO.  SAMPLE-0
 460   003110 77  HASH-TOTAL-COUNTER-ANNUAL    PICTURE 9%6¤V99 VALUE IS ZERO.  SAMPLE-3
 470   003120 77  WEEKLY-PAY                   PICTURE 999V99.                 SAMPLE-3
 480   003130 77  MONTHLY-PAY                  PICTURE 9999V99.                SAMPLE-3
 490   003140 77  ANNUAL-PAY                   PICTURE 9%5¤V99.                SAMPLE-3
 500   003150 CONSTANT SECTION.                                                SAMPLE-3
 510   003160 77  HASH-TOTAL-OF-WEEKLY-PAY   PICTURE 9%6¤V99 VALUE 008826.69.  SAMPLE-3
 520   003170 77  HASH-TOTAL-OF-MONTHLY-PAY  PICTURE 9%6¤V99 VALUE 038250.00.  SAMPLE-3
 530   003180 77  HASH-TOTAL-OF-ANNUAL-PAY   PICTURE 9%6¤V99 VALUE 459000.00.  SAMPLE-3
 540   004010 PROCEDURE DIVISION.                                              SAMPLE-3
 550   004020 START.                                                           SAMPLE-3
 560   004030     OPEN OUTPUT SALARY-FILE.                                      SAMPLE-3
 570   004040     WRITE OUTPUT-RECORD FROM HEADING-RECORD                       SAMPLE-3
```

Figure 22. Sample Program (Part 1 of 8)

```
580  004050          BEFORE ADVANCING 2 LINES.                                SAMPLE-3
590  004060      PERFORM CALCULATIONS                                         SAMPLE-3
600  004070          VARYING MONTHLY-PAY                                      SAMPLE-3
610  004080                  FROM 500                                         SAMPLE-3
620  004090                  BY    10                                         SAMPLE-3
630  004100                  UNTIL MONTHLY-PAY IS GREATER THAN 1000.          SAMPLE-3
640  004110 TEST-HASH-TOTALS.                                                SAMPLE-3
650  004120      IF  HASH-TOTAL-COUNTER-WEEKLY  # HASH-TOTAL-OF-WEEKLY-PAY   SAMPLE-3
660  004130      AND HASH-TOTAL-COUNTER-MONTHLY # HASH-TOTAL-OF-MONTHLY-PAY  SAMPLE-3
670  004140      AND HASH-TOTAL-COUNTER-ANNUAL  # HASH-TOTAL-OF-ANNUAL-PAY   SAMPLE-3
680  004150          MOVE CORRECT-MESSAGE TO OUTPUT-RECORD                    SAMPLE-3
690  004160      OTHERWISE                                                    SAMPLE-3
700  004170          MOVE INCORRECT-MESSAGE TO OUTPUT-RECORD.                 SAMPLE-3
710  004180      WRITE OUTPUT-RECORD                                          SAMPLE-3
720  004190          AFTER ADVANCING 2 LINES.                                 SAMPLE-3
730  004200      CLOSE SALARY-FILE.                                           SAMPLE-3
740  004210      STOP RUN.                                                    SAMPLE-3
750  005010 CALCULATIONS.                                                    SAMPLE-3
760  005020      COMPUTE WEEKLY-PAY #  3 * MONTHLY-PAY / 13.                  SAMPLE-3
770  005030      COMPUTE ANNUAL-PAY # 12 * MONTHLY-PAY.                       SAMPLE-3
780  005040      MOVE WEEKLY-PAY  TO WEEKLY-DETAIL-LINE.                      SAMPLE-3
790  005050      MOVE MONTHLY-PAY TO MONTHLY-DETAIL-LINE.                     SAMPLE-3
800  005060      MOVE ANNUAL-PAY  TO ANNUAL-DETAIL-LINE.                      SAMPLE-3
810  005070      ADD  WEEKLY-PAY  TO HASH-TOTAL-COUNTER-WEEKLY.               SAMPLE-3
820  005080      ADD  MONTHLY-PAY TO HASH-TOTAL-COUNTER-MONTHLY.              SAMPLE-3
830  005090      ADD  ANNUAL-PAY  TO HASH-TOTAL-COUNTER-ANNUAL.               SAMPLE-3
840  005100      WRITE OUTPUT-RECORD FROM SALARY-RECORD.                      SAMPLE-3
```

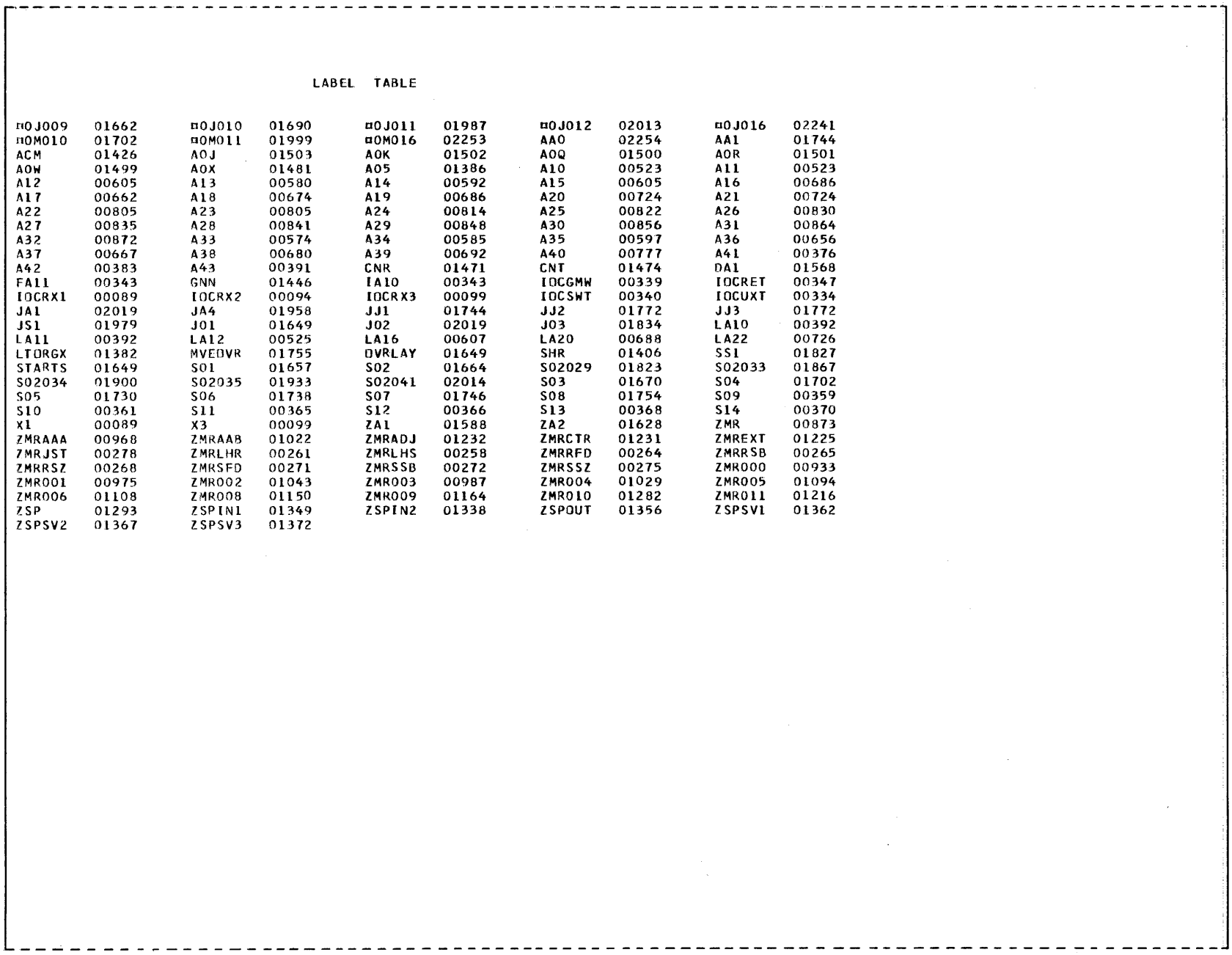
Figure 22. Sample Program (Part 2 of 8)



```
 TYPE   NAME    SOURCE

FILE     A10  SALARY-FILE
 REC     A11  OUTPUT-RECORD
  DATA   A12  SALARY-RECORD
  DATA   A13  WEEKLY-DETAIL-LINE
  DATA   A14  MONTHLY-DETAIL-LINE
  DATA   A15  ANNUAL-DETAIL-LINE
  DATA   A16  HEADING-RECORD
  DATA   A17  WEEKLY-HEADING-LINE
  DATA   A18  MONTHLY-HEADING-LINE
  DATA   A19  ANNUAL-HEADING-LINE
  DATA   A20  CORRECT-MESSAGE
  DATA   A21  TABLE-IS-CORRECT
  DATA   A22  INCORRECT-MESSAGE
  DATA   A23  TABLE-IS-NOT-CORRECT
  DATA   A24  HASH-TOTAL-COUNTER-WEEKLY
  DATA   A25  HASH-TOTAL-COUNTER-MONTHLY
  DATA   A26  HASH-TOTAL-COUNTER-ANNUAL
  DATA   A27  WEEKLY-PAY
  DATA   A28  MONTHLY-PAY
  DATA   A29  ANNUAL-PAY
  DATA   A30  HASH-TOTAL-OF-WEEKLY-PAY
  DATA   A31  HASH-TOTAL-OF-MONTHLY-PAY
  DATA   A32  HASH-TOTAL-OF-ANNUAL-PAY
PROC     J01  START
PROC     J02  CALCULATIONS
PROC     J03  TEST-HASH-TOTALS
                                    END OF COMPILATION
```

Figure 22. Sample Program (Part 3 of 8)



LABEL TABLE

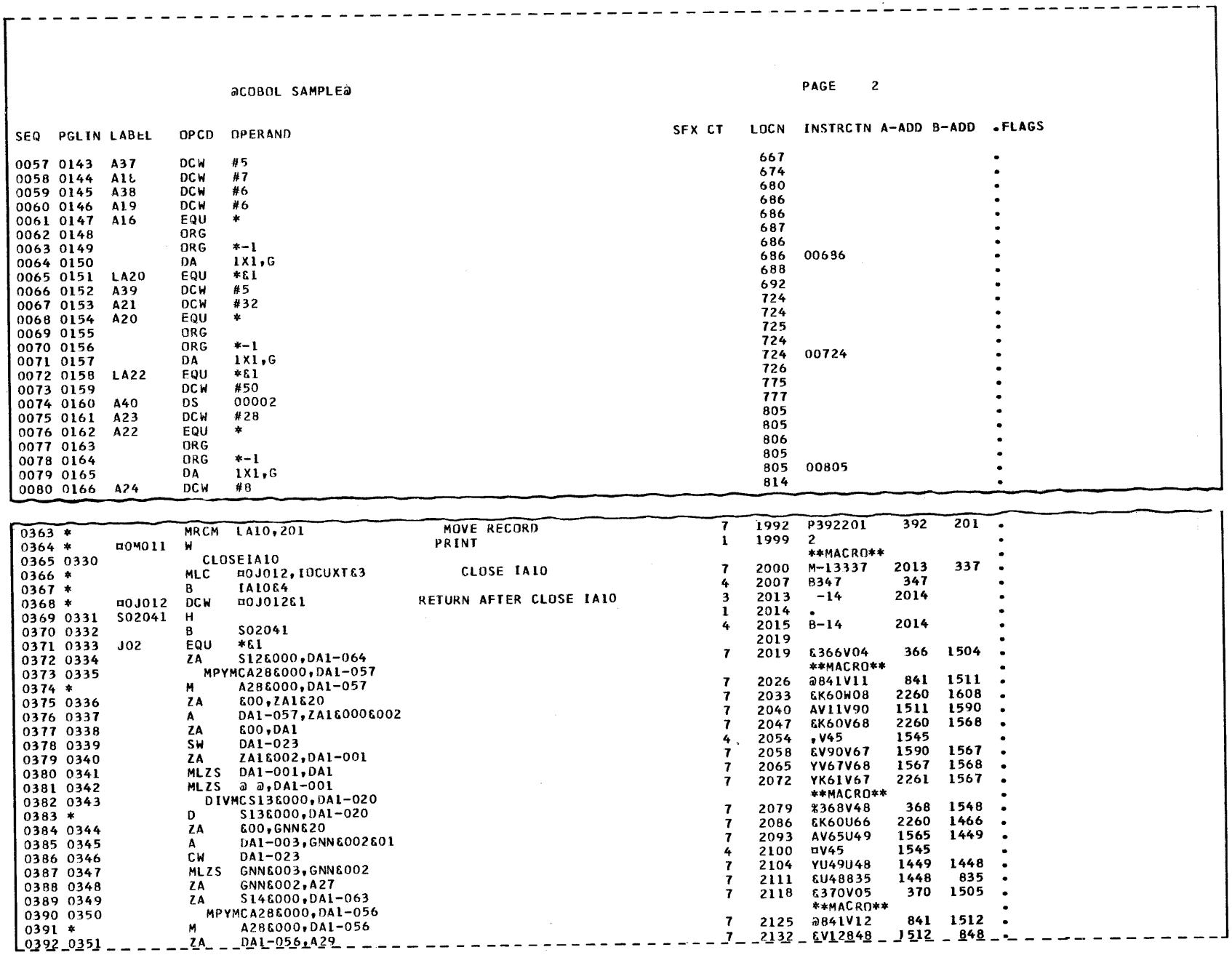
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ¤0J009 | 01662 | ¤0J010 | 01690 | ¤0J011 | 01987 | ¤0J012 | 02013 | ¤0J016 | 02241 |
| ¤0M010 | 01702 | ¤0M011 | 01999 | ¤0M016 | 02253 | AA0 | 02254 | AA1 | 01744 |
| ACM | 01426 | AOJ | 01503 | AOK | 01502 | AOQ | 01500 | AOR | 01501 |
| AOW | 01499 | AOX | 01481 | A05 | 01386 | A10 | 00523 | A11 | 00523 |
| A12 | 00605 | A13 | 00580 | A14 | 00592 | A15 | 00605 | A16 | 00686 |
| A17 | 00662 | A18 | 00674 | A19 | 00686 | A20 | 00724 | A21 | 00724 |
| A22 | 00805 | A23 | 00805 | A24 | 00814 | A25 | 00822 | A26 | 00830 |
| A27 | 00835 | A28 | 00841 | A29 | 00848 | A30 | 00856 | A31 | 00864 |
| A32 | 00872 | A33 | 00574 | A34 | 00585 | A35 | 00597 | A36 | 00656 |
| A37 | 00667 | A38 | 00680 | A39 | 00692 | A40 | 00777 | A41 | 00376 |
| A42 | 00383 | A43 | 00391 | CNR | 01471 | CNT | 01474 | DA1 | 01568 |
| FA11 | 00343 | GNN | 01446 | IA10 | 00343 | IOCGMW | 00339 | IOCRET | 00347 |
| IOCRX1 | 00089 | IOCRX2 | 00094 | IOCRX3 | 00099 | IOCSWT | 00340 | IOCUXT | 00334 |
| JA1 | 02019 | JA4 | 01958 | JJ1 | 01744 | JJ2 | 01772 | JJ3 | 01772 |
| JS1 | 01979 | J01 | 01649 | J02 | 02019 | J03 | 01834 | LA10 | 00392 |
| LA11 | 00392 | LA12 | 00525 | LA16 | 00607 | LA20 | 00688 | LA22 | 00726 |
| LTORGX | 01382 | MVEOVR | 01755 | OVRLAY | 01649 | SHR | 01406 | SS1 | 01827 |
| STARTS | 01649 | S01 | 01657 | S02 | 01664 | S02029 | 01823 | S02033 | 01867 |
| S02034 | 01900 | S02035 | 01933 | S02041 | 02014 | S03 | 01670 | S04 | 01702 |
| S05 | 01730 | S06 | 01738 | S07 | 01746 | S08 | 01754 | S09 | 00359 |
| S10 | 00361 | S11 | 00365 | S12 | 00366 | S13 | 00368 | S14 | 00370 |
| X1 | 00089 | X3 | 00099 | ZA1 | 01588 | ZA2 | 01628 | ZMR | 00873 |
| ZMRAAA | 00968 | ZMRAAB | 01022 | ZMRADJ | 01232 | ZMRCTR | 01231 | ZMREXT | 01225 |
| ZMRJST | 00278 | ZMRLHR | 00261 | ZMRLHS | 00258 | ZMRRFD | 00264 | ZMRRSB | 00265 |
| ZMRRSZ | 00268 | ZMRSFD | 00271 | ZMRSSB | 00272 | ZMRSSZ | 00275 | ZMR000 | 00933 |
| ZMR001 | 00975 | ZMR002 | 01043 | ZMR003 | 00987 | ZMR004 | 01029 | ZMR005 | 01094 |
| ZMR006 | 01108 | ZMR008 | 01150 | ZMR009 | 01164 | ZMR010 | 01282 | ZMR011 | 01216 |
| ZSP | 01293 | ZSPIN1 | 01349 | ZSPIN2 | 01338 | ZSPOUT | 01356 | ZSPSV1 | 01362 |
| ZSPSV2 | 01367 | ZSPSV3 | 01372 | | | | | | |

Figure 22. Sample Program (Part 4 of 8)

```
                          @COBOL SAMPLE@                                                PAGE    1

SEQ  PGLIN LABEL    OPCD  OPERAND                                  SFX CT   LOCN  INSTRCTN A-ADD B-ADD  .FLAGS
0001 0102          JOB   @COBOL SAMPLE@                                                              .
0002 0103          CTL   1 110     L     03701D                                                     .
0003 0104*           DIOCS                                                                        .
0004 0105* DIOCSORG       00334                                                                    .
0005 0106* IODEVICES      PRINTER                                                                  .
0006 *             ORG   87                                                 087                   .
0007 *      IOCRX1 DCW   @   @                                        3     089                   .
0008 *             DC    @  @                                        2     091                   .
0009 *      IOCRX2 DCW   @   @                                        3     094                   .
0010 *             DC    @  @                                        2     096                   .
0011 *      IOCRX3 DCW   @   @                                        3     099                   .
0012 *             ORG   00334                                              334                   .
0013 *      IOCUXT B     0                          UNIVERSAL EXIT    4     334   B000      0     .
0014 *             DCW   @ @,G    BLANK                              1     338                   .
0015 *      IOCGMW EQU   *    GROUP MARK - WORD MARK                        339                   .
0016 *      IOCSWT DC    @ @                                          1     340                   .
0017 0107*           DTF  IA10                                                                    .
0018 0108* FILETYPE      PRINTER                                                                  .
0019 0109* IOAREAS      LA10                                                                     .
0020 *             DCW   @.9@                                        2     342                   .
0021 *      IA10   B     IOCUXT                                      4     343   B334    334     .
0022 *      IOCRET BIN   IA10-2,*                TEST PRINTER ERROR 5     347   B341*   341     .
0023 *             BIN   IOCUXT,                                     5     352   B334    334     .
0024 0110            MACOP,1                                                 **MACRO**             .
0025 0111  S09     DCW   @500@                                       3     359                   .
0026 0112  S10     DCW   @10@                                        2     361                   .
0027 0113  S11     DCW   @1000@                                      4     365                   .
0028 0114  S12     DCW   @3@                                         1     366                   .
0029 0115  S13     DCW   @13@                                        2     368                   .
0030 0116  S14     DCW   @12@                                        2     370                   .
0031 0117  A41     DCW   @  0. 0@                                    6     376                   .
0032 0118  A42     DCW   @   0. 0@                                   7     383                   .
0033 0119  A43     DCW   @    0. 0@                                  8     391                   .
0034 0120  LA10    EQU   *&1                                                392                   .
0035 0121  LA11    EQU   *&1                                                392                   .
0036 0122  FA11    EQU   IA10                                               343                   .
0037 0123          DCW   #50                                                441                   .
0038 0124  A11     DS    00082                                              523                   .
0039 0125          ORG                                                      524                   .
0040 0126          ORG   *-1                                                523                   .
0041 0127          DA    1X1,G                                              523   00523           .
0042 0128  A10     EQU   A11                                                523                   .
0043 0129  LA12    EQU   *&1                                                525                   .
0044 0130  A33     DCW   #50                                                574                   .
0045 0131  A13     DCW   #6                                                 580                   .
0046 0132  A34     DCW   #5                                                 585                   .
0047 0133  A14     DCW   #7                                                 592                   .
0048 0134  A35     DCW   #5                                                 597                   .
0049 0135  A15     DCW   #8                                                 605                   .
0050 0136  A12     EQU   *                                                  605                   .
0051 0137          ORG                                                      606                   .
0052 0138          ORG   *-1                                                605                   .
0053 0139          DA    1X1,G                                              605   00605           .
0054 0140  LA16    EQU   *&1                                                607                   .
0055 0141  A36     DCW   #50                                                656                   .
0056 0142  A17     DCW   #6                                                 562                   .
```

Figure 22. Sample Program (Part 5 of 8)



```
                        @COBOL SAMPLE@                                                    PAGE    2

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

0057 0143  A37      DCW   #5                                                 667                        .
0058 0144  A18      DCW   #7                                                 674                        .
0059 0145  A38      DCW   #6                                                 680                        .
0060 0146  A19      DCW   #6                                                 686                        .
0061 0147  A16      EQU   *                                                  686                        .
0062 0148           ORG                                                      687                        .
0063 0149           ORG   *-1                                                686                        .
0064 0150           DA    1X1,G                                              686  00686                 .
0065 0151  LA20     EQU   *&1                                                688                        .
0066 0152  A39      DCW   #5                                                 692                        .
0067 0153  A21      DCW   #32                                                724                        .
0068 0154  A20      EQU   *                                                  724                        .
0069 0155           ORG                                                      725                        .
0070 0156           ORG   *-1                                                724                        .
0071 0157           DA    1X1,G                                              724  00724                 .
0072 0158  LA22     EQU   *&1                                                726                        .
0073 0159           DCW   #50                                                775                        .
0074 0160  A40      DS    00002                                              777                        .
0075 0161  A23      DCW   #28                                                805                        .
0076 0162  A22      EQU   *                                                  805                        .
0077 0163           ORG                                                      806                        .
0078 0164           ORG   *-1                                                805                        .
0079 0165           DA    1X1,G                                              805  00805                 .
0080 0166  A24      DCW   #8                                                 814                        .

0363 *              MRCM  LA10,201          MOVE RECORD                 7    1992  P392201   392   201  .
0364 *      ¤OM011  W                       PRINT                       1    1999  2                      .
0365 0330             CLOSEIA10                                                   **MACRO**              .
0366 *              MLC   ¤OJ012,IOCUXT&3       CLOSE IA10              7    2000  M-13337   2013   337  .
0367 *              B     IA10&4                                        4    2007  B347      347          .
0368 *      ¤OJ012  DCW   ¤OJ012&1        RETURN AFTER CLOSE IA10       3    2013   -14      2014         .
0369 0331  S02041   H                                                   1    2014  .                      .
0370 0332           B     S02041                                        4    2015  B-14      2014         .
0371 0333  J02      EQU   *&1                                                 2019                       .
0372 0334           ZA    S12&000,DA1-064                               7    2019  &366V04   366   1504 .
0373 0335             MPYMCA28&000,DA1-057                                      **MACRO**              .
0374 *              M     A28&000,DA1-057                               7    2026  @841V11   841   1511 .
0375 0336           ZA    &00,ZA1&20                                    7    2033  &K60W08   2260  1608 .
0376 0337           A     DA1-057,ZA1&000&002                           7    2040  AV11V90   1511  1590 .
0377 0338           ZA    &00,DA1                                       7    2047  &K60V68   2260  1568 .
0378 0339           SW    DA1-023                                       4    2054  ,V45      1545         .
0379 0340           ZA    ZA1&002,DA1-001                               7    2058  &V90V67   1590  1567 .
0380 0341           MLZS  DA1-001,DA1                                   7    2065  YV67V68   1567  1568 .
0381 0342           MLZS  @ @,DA1-001                                   7    2072  YK61V67   2261  1567 .
0382 0343             DIVMCS13&000,DA1-020                                      **MACRO**              .
0383 *              D     S13&000,DA1-020                               7    2079  %368V48   368   1548 .
0384 0344           ZA    &00,GNN&20                                    7    2086  &K60U66   2260  1466 .
0385 0345           A     DA1-003,GNN&002&01                            7    2093  AV65U49   1565  1449 .
0386 0346           CW    DA1-023                                       4    2100  ¤V45      1545         .
0387 0347           MLZS  GNN&003,GNN&002                               7    2104  YU49U48   1449  1448 .
0388 0348           ZA    GNN&002,A27                                   7    2111  &U48835   1448  835  .
0389 0349           ZA    S14&000,DA1-063                               7    2118  &370V05   370   1505 .
0390 0350             MPYMCA28&000,DA1-056                                      **MACRO**              .
0391 *              M     A28&000,DA1-056                               7    2125  @841V12   841   1512 .
0392 0351           ZA    DA1-056,A29                                   7    2132  &V12848   1512  848  .
```

Figure 22. Sample Program (Part 6 of 8)

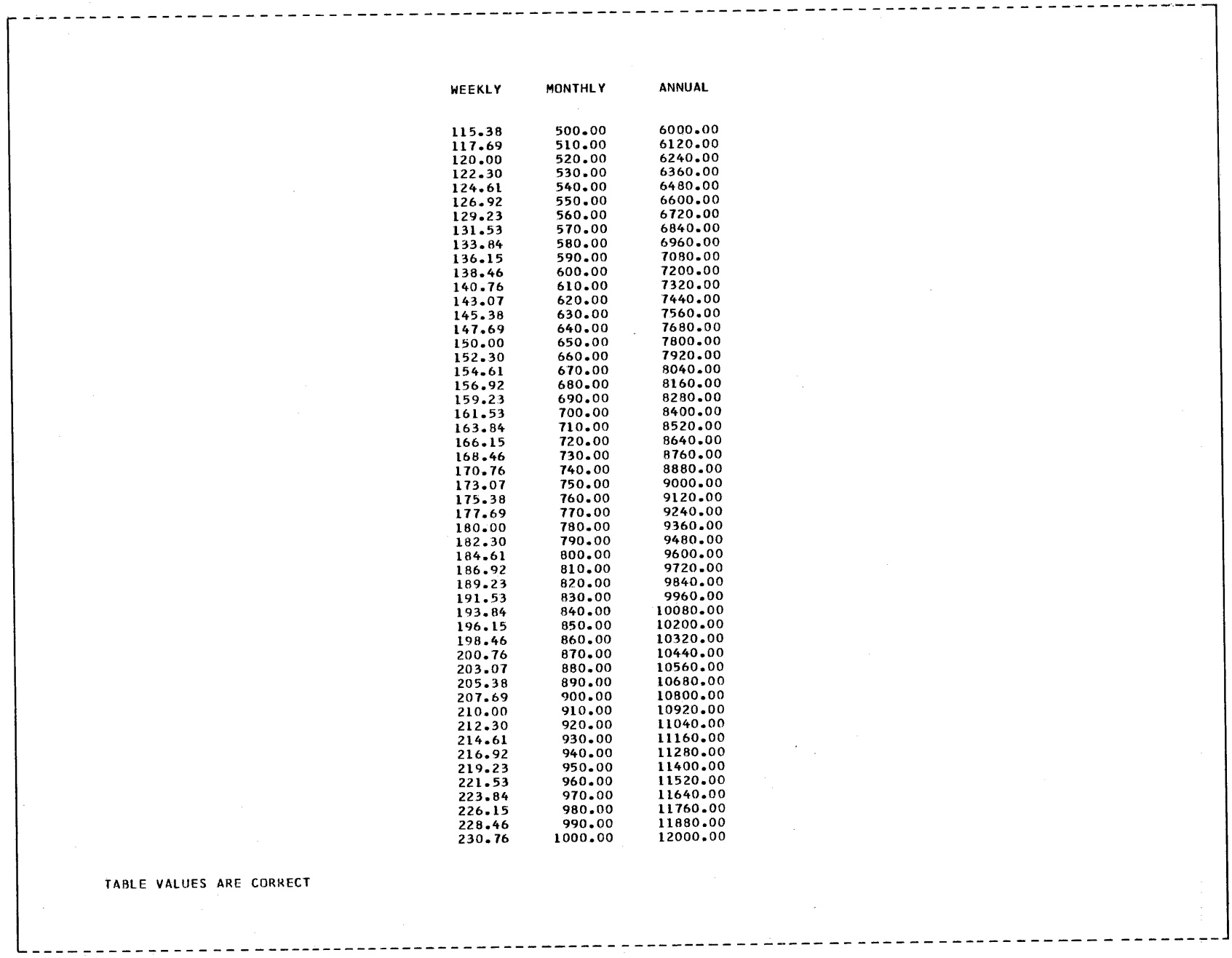
Figure 22. Sample Program (Part 7 of 8)

```
                                @COBOL SAMPLE@                                                    PAGE    8

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

0393 0352          MLCWA A41,A13                                                   7   2139  L376580    376   580  .
0394 0353          MCE   A27&000,A13                                               7   2146  E835580    835   580  .
0395 0354          SW    A13-006&01                                                4   2153  ,575       575        .
0396 0355          MLCWA A42,A14                                                   7   2157  L383592    383   592  .
0397 0356          MCE   A28&000,A14                                               7   2164  E841592    841   592  .
0398 0357          SW    A14-007&01                                                4   2171  ,586       586        .
0399 0358          MLCWA A43,A15                                                   7   2175  L391605    391   605  .
0400 0359          MCE   A29&000,A15                                               7   2182  E848605    848   605  .
0401 0360          SW    A15-008&01                                                4   2189  ,598       598        .
0402 0361          A     A27&000,A24&000                                           7   2193  A835814    835   814  .
0403 0362          A     A28&000,A25&000                                           7   2200  A841822    841   822  .
0404 0363          A     A29&000,A26&000                                           7   2207  A848830    848   830  .
0405 0364          B     ZMR                                                       4   2214  B873       873        .
0406 0365          DCW   A11                                                       3   2220   523       523        .
0407 0366          DCW   0                                                       1   2221                          .
0408 0367          DCW   132                                                     3   2224                          .
0409 0368          DCW   A12                                                     3   2227   605       605        .
0410 0369          DCW   0                                                       1   2228                          .
0411 0370          DCW   081                                                     3   2231                          .
0412 0371          DCW   @00L@                                                   3   2234                          .
0413 0372            PUT  LA10,IA10                                                    **MACRO**                   .
0414 *             B     ¤0J016                                                  4   2235  BK41       2241        .
0415 *             DCW   @.9@          PRINTER ERROR - PRESS START               2   2240                          .
0416 *      ¤0J016 BIN   *-6,*            TEST PRINTER ERROR                    5   2241  BK39*      2239        .
0417 *             MRCM  LA10,201           MOVE RECORD                        7   2246  P392201     392   201  .
0418 *      ¤0M016 W                        PRINT                              1   2253  2                         .
0419 0373   AA0    NOP   AA1                                                   4   2254  NX44       1744        .
0420 0374          END   STARTS                                                      W49        1649        .
0421               LTRL  @R@                                                   1   2258                          .
0422               LTRL  &00                                                   2   2260                          .
0423               LTRL  @ @                                                   1   2261                          .
0424               LTRL  @N@                                                   1   2262                          .
                       END OF LISTING                                          NO SEQUENCE ERRORS
```

```
CORE LOAD HEADER-@COBOL SAMPLE@                                                , ID-
CORE LOAD OUTPUT COMPLETE ON 1311 UNIT 0, START 000100, END 000152
```



| WEEKLY | MONTHLY | ANNUAL |
|---|---|---|
| 115.38 | 500.00 | 6000.00 |
| 117.69 | 510.00 | 6120.00 |
| 120.00 | 520.00 | 6240.00 |
| 122.30 | 530.00 | 6360.00 |
| 124.61 | 540.00 | 6480.00 |
| 126.92 | 550.00 | 6600.00 |
| 129.23 | 560.00 | 6720.00 |
| 131.53 | 570.00 | 6840.00 |
| 133.84 | 580.00 | 6960.00 |
| 136.15 | 590.00 | 7080.00 |
| 138.46 | 600.00 | 7200.00 |
| 140.76 | 610.00 | 7320.00 |
| 143.07 | 620.00 | 7440.00 |
| 145.38 | 630.00 | 7560.00 |
| 147.69 | 640.00 | 7680.00 |
| 150.00 | 650.00 | 7800.00 |
| 152.30 | 660.00 | 7920.00 |
| 154.61 | 670.00 | 8040.00 |
| 156.92 | 680.00 | 8160.00 |
| 159.23 | 690.00 | 8280.00 |
| 161.53 | 700.00 | 8400.00 |
| 163.84 | 710.00 | 8520.00 |
| 166.15 | 720.00 | 8640.00 |
| 168.46 | 730.00 | 8760.00 |
| 170.76 | 740.00 | 8880.00 |
| 173.07 | 750.00 | 9000.00 |
| 175.38 | 760.00 | 9120.00 |
| 177.69 | 770.00 | 9240.00 |
| 180.00 | 780.00 | 9360.00 |
| 182.30 | 790.00 | 9480.00 |
| 184.61 | 800.00 | 9600.00 |
| 186.92 | 810.00 | 9720.00 |
| 189.23 | 820.00 | 9840.00 |
| 191.53 | 830.00 | 9960.00 |
| 193.84 | 840.00 | 10080.00 |
| 196.15 | 850.00 | 10200.00 |
| 198.46 | 860.00 | 10320.00 |
| 200.76 | 870.00 | 10440.00 |
| 203.07 | 880.00 | 10560.00 |
| 205.38 | 890.00 | 10680.00 |
| 207.69 | 900.00 | 10800.00 |
| 210.00 | 910.00 | 10920.00 |
| 212.30 | 920.00 | 11040.00 |
| 214.61 | 930.00 | 11160.00 |
| 216.92 | 940.00 | 11280.00 |
| 219.23 | 950.00 | 11400.00 |
| 221.53 | 960.00 | 11520.00 |
| 223.84 | 970.00 | 11640.00 |
| 226.15 | 980.00 | 11760.00 |
| 228.46 | 990.00 | 11880.00 |
| 230.76 | 1000.00 | 12000.00 |

TABLE VALUES ARE CORRECT

Figure 22. Sample Program (Part 8 of 8)

# Index


ACEPT Macro ..... 9
ALCOM Macro ..... 9
ASGN Cards ..... 8, 25
Assembler ..... 5
Assembly ..... 5
Autocoder Assembler Program ..... 6, 9
Autocoder Library ..... 6, 37, 38
Autocoder System ..... 37, 38
Autocoder Text ..... 21, 22

Batched Files ..... 5, 28
Building a COBOL System ..... 36, 38

Card Boot ..... 5
Changing File Assignments ..... 25
Clear Disk ..... 38, 39
COBOL Compiler ..... 6, 8
COBOL Compiler Output ..... 13
COBOL Diagnostic Messages ..... 13
COBOL Macros ..... 6, 9, 38, 41
COBOL RUN ..... 7, 21
COBOL RUN THRU AUTOCODER ..... 7, 21
COBOL RUN THRU OUTPUT ..... 7, 22, 36
COBOL RUN THRU EXECUTION ..... 7, 23
COBOL Sample Program ..... 38
COBOL System
- Building a ..... 36, 38
- Components of ..... 6
- Deck Description ..... 36, 37
- Definition of ..... 6
- Features of ..... 6
- Updating a ..... 42

COBOL Update ..... 38, 41
Condensed Loader Format ..... 29
CONTROL ASGN ..... 24, 42
Control Cards ..... 7
CONTROL File ..... 7, 27
CORELOAD ASGN Card ..... 25, 42
CORELOAD File ..... 7
Cross Reference List ..... 21

Deck Description ..... 36, 37
Definition of Key Terms ..... 5
Diagnostic Messages ..... 15, 16, 17, 18, 19
Dictionary ..... 13
DIVDE Macro ..... 9
DSPLY Macro ..... 9

EDIT1 Macro ..... 9
EXPIN Macro ..... 9
EXPNI Macro ..... 9
External Files ..... 7

FGCOM Macro ..... 9
File Considerations ..... 27
Files, Batched ..... 27
Files, Logical (See Logical Files) .....

GOTOD Macro ..... 11

HALT Card ..... 8
Halts and Messages ..... 30, 31, 32, 33, 34, 35

INPUT ASGN Card ..... 24, 42
IFALP Macro ..... 11
IFNUM ..... 11
INDIX Macro ..... 11
INPUT FILE ..... 7, 27
Internal Files ..... 7
IOCS ..... 6, 9

Jobs
- Definition of ..... 5
- Performing ..... 28
- Preparing ..... 20
- Processor ..... 20
- Stacked ..... 28
- Update ..... 20, 41

Label Table ..... 21, 22, 23
LIBRARY ASGN Card ..... 24, 42
Library, Autocoder ..... 37, 38
LIBRARY FILE ..... 7
LIST ASGN Card ..... 24, 42
LIST File ..... 7, 13, 27
Listing, Program ..... 13
Load-and-Go ..... 23
Loader, Condensed ..... 29
Loading Object Programs ..... 29
Logical Files
- Assumed Assignments ..... 24, 42
- Batched ..... 7, 28
- Changing Assignments ..... 24, 25, 26, 27
- Considerations ..... 25, 27
- Definition of ..... 5
- External ..... 7
- Function of ..... 6
- Internal ..... 7
- Operation ..... 7
- Residence ..... 7

Machine Operator ..... 2
Machine Requirements ..... 5
Marking Program ..... 37
MESSAGE ASGN Card ..... 24, 42
MESSAGE File ..... 7, 27
MULTY Macro ..... 11
MVALL Macro ..... 11
MVFTR Macro ..... 12

Name-Associated Diagnostics ..... 13
NOTE Card ..... 8

Object Time ..... 5
Operating Procedures ..... 20
Operation ..... 5
Operation Files ..... 7
OUTPUT ASGN Card ..... 24, 42
OUTPUT File ..... 7, 27
Output, Listed ..... 13, 21, 22, 23
Output Options ..... 21, 22, 23

PAUSE Card ..... 8
Phase Descriptions ..... 44
Preparing ASGN Cards ..... 25
Processor Jobs ..... 20, 21, 22, 23
Program, Source ..... 20
Program, System Control ..... 6

Qualification Table ..... 13

Related Information ... 2
Residence File ... 7
RUN Card ... 7

Sample Program ... 42
Self-Loading Format ... 29
Source Deck, Composition of ... 20
Source Program ... 20
Source Program Listing ... 13
Source Statement Diagnostics ... 13
Split Cylinder ... 27
SPLIT Macro ... 12
Stack
  Definition of ... 5
  Preparation of ... 28
  Running a ... 29
SUBS1 Macro ... 12
SUBS2 Macro ... 12
SUBS3 Macro ... 12
System ... 5
SYSTEM ASGN Card ... 24, 42
System Control Modification ... 38, 39
System Control Program ... 6
SYSTEM File ... 7, 27

Text, Autocoder ... 21, 22
Timing Considerations ... 27

UPDAT Card ... 8
Update ... 8
Update Jobs ... 20
Updating a COBOL System ... 42
User-Assignments (Logical Files) ... 24
Using ASGN Cards ... 28

WORK ASGN Card ... 24, 42
WORK1 File ... 7, 27
WORK2 File ... 7, 27
WORK3 File ... 7, 27
WORK4 File ... 7, 27
WORK5 File ... 7, 13, 27
Write File-Protected Addresses ... 38, 40

XAMIN Macro ... 12

IBM 1401/1440/1460 Printed in U. S. A. C24-3242-2

IBM

International Business Machines Corporation

Data Processing Division

112 East Post Road, White Plains, N. Y. 10601