File Number 1401-24
Form C24-3146-3

IBM Systems Reference Library

COBOL (on Tape) Operating Procedures IBM 1401

Program Number 1401-CB-070

This publication describes the operating procedures used to assemble a machine-language program from a source program written in 1401 COBOL language. It also describes the operating procedures for producing and modifying the COBOL tapes, the phases that make up the COBOL processor program, and lists diagnostic and error messages in detail.

Fourth Edition

This is a major revision of, and obsoletes C24-3146-1, C24-3146-2, and Technical Newsletters N24-0244, N21-0032, and N21-0048. The section headed *Placing Prescan Diagnostics on the COBOL Processor System Tape* has been deleted. Changes are indicated by a vertical line at the left of the affected text.

Significant changes or additions to the specifications contained in this publication will be reported in subsequent revisions or Technical Newsletters.

Contents

Machine Requirements 5

Operating Functions 5

Copying the PID Tape 6
Creating COBOL Processor and Extraction Tapes 6
Printing COBOL Phase Listings 6
Punching COBOL Subroutines, Sample Program 6
Placing COBOL Subroutines in the Autocoder Library 6
Duplicating the 1401 COBOL Processor System Tape 7
Performing a Prescan Diagnostic Run 7
Prescan Run Output 7
Compiling COBOL Source Programs (COBOL to Autocoder) 8
Assembling Autocoder Output (Autocoder to Machine Language) 9
Loading the Autocoder Object Program 9
Obtaining Storage and Tape Prints for APAR's 10
COBOL Dictionary 10

COBOL Diagnostic Usage 10

Prescan Diagnostics 10
COBOL Processor Diagnostics 12
Diagnostic Reference List 14
COBOL Tape Print Program 21

COBOL Phase Description 22

COBOL Subroutines 27
Halts and Messages 27

Sample Program: Table of Salaries 29

IBM 1401 COBOL (on Tape) Operating Procedures

Machine Requirements

The 1401 COBOL processor can compile an object program for any IBM 1401 system that has at least 4,000 positions of core storage. To process the COBOL source program, the 1401 must have at least:

- 4,000 positions of core storage
- Four IBM magnetic-tape units
- IBM 1403 Printer, Model 2
- IBM 1402 Card Read-Punch
- Advanced Programming Feature
- High-Low-Equal Compare Feature
- Sense switches.

A minimum of 8000 positions of core storage is required if a prescan diagnostic run (described later in this publication) is to be performed.

The 1401 on which the object program is to be executed must have:

1. Sufficient core storage to contain the program produced by the COBOL processor. If the object program requires more than the available core-storage capacity, either the program must be executed in sections (overlays) or the job must be divided into multiple runs. This is a significant consideration when COBOL is to be used on systems with 4000 positions of main storage.
2. The object machine must have the input and output units defined in the FILE-CONTROL paragraph.
3. Advanced Programming Feature.
4. High-Low-Equal Compare Feature.
5. Sense switches when they are referred to in the SPECIAL-NAMES section.
6. Multiply-Divide Feature if any of these entries appears in the Procedure Division of the COBOL source program:
   a. MULTIPLY verb
   b. DIVIDE verb
   c. COMPUTE verb when either /, *, or ** is used as the operator.

Operating Functions

The COBOL system contains several processing functions that facilitate compiling and interpreting COBOL programs. These functions are:

- Copying the PID tape.
- Creating the COBOL processor and extraction tapes.
- Printing COBOL phase listings.
- Punching COBOL subroutines and sample programs.
- Placing COBOL subroutines in the Autocoder library.
- Compiling COBOL source programs (COBOL to Autocoder).
- Assembling COBOL output (Autocoder to machine language).
- Loading the object program.
- Duplicating the 1401 COBOL processor system tape.
- Obtaining storage and tape prints for APARS.

The 1401 COBOL processor is a self-loading magnetic tape. It should be labeled *1401 COBOL Processor System* with the current version number and modification level of the program.

1401 COBOL Extraction System is a self-loading program written on magnetic tape. This program is used to print a listing of the COBOL processor system, to punch the COBOL subroutines deck used to update an Autocoder system tape, to punch the sample program: *Table of Salaries* (see *Operating Functions*).

The *Table of Salaries* in the sample problem is an example of how a computer can be directed by using COBOL language to compute a set of values and list them in tabular form on the printer. The sample program is written in COBOL source language and must be compiled and assembled before an object run can be made.

To assemble the compiled output of the 1401 COBOL processor, the user's 1401 Autocoder system tape (the latest version and modification level) should be used. This tape must have been updated with the COBOL subroutines deck and should contain the specific IOCS appropriate to the user's needs (see *Placing COBOL Subroutines in the Autocoder Library*).

Operating instructions for using each of these items are included in this publication.

In addition to the material presented here, the programmer should be familiar with the material given in the 1401 Systems Reference Library (SRL) publication that describes the IBM 1401 configuration for which the program is written. These are listed with form numbers in the *IBM 1401 Bibliography* (A24-1495).

The user must be familiar with the information contained in *COBOL (on Tape) Specifications for IBM 1401* (C24-1492) and the *IBM COBOL General Information Manual* (F28-8053). Together they provide programmers with sufficient information to enable them to write programs in the 1401 COBOL. To use the available features of the COBOL language and the COBOL system effectively, the user should have a thorough understanding of this material.

Copying the PID Tape

The tape labeled *1401 COBOL PID Tape,* sent by the IBM Program Information Department, can be copied as follows:

1. Mount the PID tape on tape unit 1.
2. Ready tape 2.
3. Set sense switch G on (up).
4. Press CHECK RESET.
5. Press START RESET.
6. Press TAPE LOAD.
7. At the end of a successful run, a programmed halt occurs with the A-address register containing 003.
8. Remove the tapes from units 1 and 2.
9. File-protect the tape from unit 2 and label it: *1401 COBOL PID Tape.*

Creating COBOL Processor and Extraction Tapes

The tape labeled *1401 COBOL PID Tape,* sent by the IBM Program Information Department, is used for this operation. This tape is a self-loading program that produces a *1401 COBOL Processor System* tape and a *1401 COBOL Extraction System* tape. To obtain these tapes:

1. Mount the PID tape on tape unit 1.
2. Ready tapes 2 and 3.
3. Press CHECK RESET.
4. Press START RESET.
5. Press TAPE LOAD.
6. At the end of a successful run, a programmed halt occurs with the A-address register containing 002.
7. Remove the tapes from the tape units and file-protect them.
8. Label the tape from unit 2: *1401 COBOL Processor System with Prescan.*
9. Label the tape from unit 3: *1401 COBOL Extraction System.*

If more than ten read or write errors are encountered for a given record while copying the PID tape or creating the COBOL processor and extraction tapes, a programmed halt occurs with the A-address register containing 004. Press the start key to reread or rewrite the record.

Printing COBOL Phase Listings

The tape labeled *1401 COBOL Extraction System,* which is extracted from the 1401 COBOL PID tape, is used for this operation. This tape contains a self-loading program that can be used to produce a symbolic listing of all phases of the 1401 COBOL system. To obtain this listing:

1. Place the *1401 COBOL Extraction System* tape on tape unit 1.
2. Press CHECK RESET and START RESET.
3. Press TAPE LOAD. The program will stop at 364.
4. Press START to produce the listing.
5. After all phases have been printed, the program halts at 600.
6. Press START RESET and START to rewind the tape, or press START to punch the subroutines and the sample program.

Punching COBOL Subroutines, Sample Program

To punch subroutines and the sample program:

1. Place the *1401 COBOL Extraction System* tape on tape unit 1.
2. Press CHECK RESET and START RESET.
3. Press TAPE LOAD. The program will stop at 364.
4. Press START RESET and START.
5. After the COBOL subroutines and the sample problem have been punched, the program will stop at 709.
6. The COBOL subroutine decks and the sample problem decks should be separated and labeled for identification as follows:
   a. The deck with the identification number 70001 punched in columns 76-80 should be labeled: *4K COBOL Subroutines.*
   b. The deck with identification number 70002 punched in columns 76-80 should be labeled: *> 4K Subroutines.*
   c. The deck with identification number 70003 punched in columns 76-80 should be labeled: *COBOL Sample Program.*

Placing COBOL Subroutines in the Autocoder Library

In this operation, an Autocoder system tape is updated to contain the routines needed to assemble the machine-language object program from the symbolic statements that the COBOL processor produces from the COBOL source program.

The Autocoder system tape used should contain the specific IOCS and macro combinations appropriate for the user. This choice should be made according to the following criteria:

1. If the user wants overlapped IOCS instructions to be generated, the Autocoder tape must contain IOCS with overlap.

2. If the user wants non-overlapped IOCS instructions to be generated by the Autocoder system, the Autocoder tape must contain IOCS without overlap.
3. If the user wants non-overlapped IOCS instructions to be generated by the COBOL system (regardless of the OBJECT-COMPUTER entry), the Autocoder tape must contain those macros that will be used by the source program while in the ENTER AUTOCODER mode.

If full label checking is used, the RDLIN macro should be included.

COBOL subroutines must be used to update the appropriate Autocoder tape for assemblies to be run on an IBM 1401 Data Processing System.

To update the Autocoder tape:

1. Place a current version of 1401 Autocoder on tape unit 1.
2. Ready a tape on tape unit 6.
3. Place either the 4K or > 4K COBOL subroutines in the card reader, depending on the size of the object machine used.
4. Turn on sense switches A and F.
5. Press START RESET.
6. Press TAPE LOAD.
7. After the Autocoder system tape has been updated, the message 1401 AUTOCODER SYSTEM COPIED ON TAPE UNIT 6 will be printed.

Duplicating the 1401 COBOL Processor System Tape

To make a copy of the 1401 COBOL processor:

1. Place the tape labeled *1401 COBOL Processor System with Prescan* on tape unit 1.
2. Ready a working tape on tape unit 2.
3. Place these two control cards in the card reader.

Card	*Columns*	*Content*
1	6-12	SYSTEMS
	16-18	RUN
2	16-18	END

4. Turn on I/O CHECK STOP.
5. Turn on sense switch A.
6. Press CHECK RESET.
7. Press START RESET.
8. Press TAPE LOAD.
9. Press START. (Because there are only two control cards, the start key must be pressed a second time to read in the END control card.)
10. After the tape has been duplicated, a programmed halt occurs; the A-address register contains 040, and the message NEW SYSTEM ON UNIT 2 is printed.

Performing a Prescan Diagnostic Run

A prescan diagnostic run, a separate and optional run, checks the COBOL source program before compilation. Input is from cards only. An END OF SOURCE card (see *Compiling COBOL Source Programs*), although it is not required and has no function in a prescan run, can be included at the end of the COBOL source program. To perform a prescan diagnostic run:

1. Mount the 1401 COBOL processor system tape on tape unit 1.
2. Mount scratch tapes on tape units 4, 5, and 6.
3. Turn on sense switch A.
4. Place the following control card ahead of the COBOL source program in the card reader:

Columns	*Content*
6-9	DIAG
16-18	RUN

NOTE: A COBOL RUN card must not be used in the prescan run.

5. Press CHECK RESET and START RESET.
6. Press TAPE LOAD.
7. Press START when the machine halts. (This causes the last card from the reader to be read, and prescan processing to begin.)

When the prescan run is complete, the machine halts. The B-address register contains 999.

Prescan Run Output

The output from the prescan diagnostic run is in printed form, and includes at most, three separate listings: a source program listing, a name listing, and a sentence-structure-error listing.

Source Program Listing: This listing contains the COBOL source program (including sequence numbers), and always appears in the prescan output. A sequence error in the listing is identified by an asterisk (*) following the sequence number.

Name Listing: This listing contains all the procedure and data names used in the source program, plus an indication of the type of each data name. The name listing always appears in the prescan output. A name that refers to more than one field (multidefined) is identified by an M between the name and its type indication. The M appears for all multidefined names, whether qualified or not, and serves as a warning that such names require qualification. A COBOL keyword, used incorrectly, is flagged by the word KEYWORD preceding the type indication. (For examples of the name-associated diagnostics, see *COBOL Diagnostic Usage.*)

Sentence-Structure-Error Listing: This listing consists of sentence-structure-error diagnostics generated during the prescan run. It appears only if errors were made in any of the COBOL sentences in the source program. (The sentence-structure-error diagnostics are more fully explained under *COBOL Diagnostic Usage.*)

Compiling COBOL Source Programs (COBOL to Autocoder)

In this operation, the COBOL source program and the COBOL processor system tape are used. The processor compiles and assembles the object program in Autocoder language. The 1401 COBOL processor prints a listing of COBOL source statements followed by the Autocoder expansion of these statements at assembly time if desired by the user. This optional source-symbolic listing is controlled by sense switch G at compilation time.

Either the card reader or a program library tape can be used to supply the source program to the COBOL processor. Two control cards regulate this process.

1. The COBOL RUN card conveys to the compiler a request for a COBOL compilation, the input option requested, and the program to be run (if the tape option was requested). The format of the COBOL RUN card is:

Columns	*Content*
6-10	COBOL
16-18	RUN
21-35	15-character identification (only required when tape option is used)
79-80	%□ (request for tape input) bb (request for card input)

2. The END OF SOURCE card follows the last card of the source program. It is required when the source program is written on tape and when the source program is stacked in the card reader. Its format is:

Columns	*Content*
1-13	END OF SOURCE

If the user desires to use the program library feature, the following requisites must be met:

1. The set of source programs must be recorded as one file, followed by a tape mark, and must be recorded as eighty-character move-mode records. (The 1401 Multiple Utility Program will create this tape.)
2. Each source program must be preceded by a COBOL RUN card and followed by an END OF SOURCE card.
3. Each COBOL RUN card must have a unique identification in columns 21-35 for the processor to select the correct source program (the %□ punches are not required in the COBOL RUN card images on tape). If the processor fails to find the programs requested, it will rewind the library tape and search it once more. After compilation, the library tape will be positioned at the next COBOL RUN card. The cards placed into the card reader may consist only of COBOL RUN cards with %□ punched in columns 79 and 80 (requesting input from the program library), a single card deck (COBOL RUN, source deck), a stack of card decks (COBOL RUN, source deck, END OF SOURCE, COBOL RUN . . .), or any combination of the above.

The following restrictions apply to the use of the COBOL RUN and END OF SOURCE cards:

1. If input is to be from the card reader, the COBOL RUN card must not have the %□ punches in columns 79 and 80.
2. If the input is from the card reader, only the last source program need not have an END OF SOURCE card.

To compile the COBOL program:

1. Place the tape labeled *1401 COBOL Processor System* on tape unit 1.

2. Ready the tapes on tape units 4, 5, and 6.
3. Ready a tape on tape unit 3 if the COBOL source symbolic listing option is used.
4. Turn on I/O CHECK STOP.
5. Turn on sense switches A and C. Turn off sense switches B, D, and E.
6. Turn on sense switch F if IOCS generation by COBOL is not desired, or if the overlap feature is used in the object program.
7. Turn on sense switch G if the COBOL source symbolic listing option is used.
8. If this is the initial compilation of a stacked job or the compilation of a single program, place the COBOL RUN card(s) and/or the source deck(s) in the card reader.

   If the program library option is used (%⌑ in columns 79 and 80 of single COBOL RUN card(s)), mount the program library tape on unit 2.
9. Press CHECK RESET.
10. Press START RESET.
11. Press TAPE LOAD.
12. Press START.
13. When card reading stops, press the start key to read in the last card.
14. At the end of a successful compilation, the messages printed are:

    IF EXTRA OUTPUT DESIRED
    B ON FOR PUNCHED AUTOCODER
    D ON FOR PRINTED LISTING
    AND PRESS START
    IF NO EXTRA OUTPUT DESIRED, PRESS START
15. After a successful compilation, the symbolic object program can be assembled. However, any errors indicated by diagnostics flagged by ** should be corrected before the symbolic object program is assembled.
16. If a symbolic listing is desired before assembly, turn on sense switch D and press START.
17. If a punched symbolic deck is desired, turn on sense switch B and press START. A symbolic listing and symbolic deck can be obtained by turning on sense switches B and D and pressing START. This sense-switch option can be used after the compilation has been completed and the message (described in item 13) has been printed.
18. When no more output is desired, turn off sense switches B and D and press START.
19. At the end of compilation, the tape on tape unit 4 will contain the symbolic program. (The symbolic program will have been punched also if the option in item 16 was used.) The message END OF COMPILATION—AUTOCODER ON TAPE 4 will be printed.

Assembling Autocoder Output (Autocoder to Machine Language)

The output from the COBOL compilation process is the object program in Autocoder language. To assemble the machine-language object program:

1. Place a current version of 1401 Autocoder (with COBOL subroutines) on tape unit 1.
2. Ready tapes on tape units 4, 5, and 6.
3. Ready a tape on tape unit 3 if the COBOL source program specified ASSIGN OBJECT PROGRAM TO TAPE.
4. If card input is used, perform these eleven steps:
   a. Turn on sense switch A.
   b. Turn on I/O CHECK STOP.
   c. Place the Autocoder program deck in the card reader.
   d. Press CHECK RESET.
   e. Press START RESET.
   f. Press TAPE LOAD.
   g. If ASSIGN OBJECT PROGRAM TO TAPE was not specified, turn on the card-read punch.
   h. Press START.
   i. When card reading stops, press START to read in the last card.
   j. At the end of assembly, the message END OF ASSEMBLY will be printed.
   k. Press START.
5. If tape input is used, perform these eight steps.
   a. Turn on sense switches A and C.
   b. Turn on I/O CHECK STOP.
   c. Press CHECK RESET.
   d. Press START RESET.
   e. Press TAPE LOAD.
   f. If ASSIGN OBJECT PROGRAM TO TAPE was not specified, turn on the card read-punch and press START.
   g. At the end of assembly, the message END OF ASSEMBLY will be printed.
   h. Press START.

Loading the Autocoder Object Program

To load the object program into the object computer:

1. If the object program is in punched-card form (instead of on tape), place the object deck in the card reader and turn on the reader.
2. If the ASSIGN OBJECT PROGRAM TO TAPE option was used in the COBOL source program, place the Auto-

coder output (the object program that was on tape unit 6 at the end of the Autocoder assembly) on tape unit 1.

3. Press CHECK RESET.
4. Press START RESET.
5. Turn on sense switch A and I/O CHECK STOP.
6. If the object program is in card form, press PROGRAM LOAD. If the object program is on tape, press TAPE LOAD.

Obtaining Storage and Tape Prints for APAR's

If difficulty is encountered during the processing of a COBOL program, and the COBOL processor is the cause of the error:

1. Set the manual address switches to 0359.
2. Set the mode switch to ALTER.
3. Press CHECK RESET.
4. Press the I-address register key.
5. Press START RESET.
6. Press START.
7. Set the mode switch to RUN.
8. Press START.
9. The contents of core storage and tapes 4, 5, and 6 will be printed.
10. After the tape prints are completed, the message TAPES 5, 6, AND 4 PRINTED will be displayed on the listing.
11. Contact your IBM Systems Engineer to submit an APAR (Authorized Programming Analysis Report).

Note: If during the tape print program bad records cannot be bypassed, branch manually to 1729.

COBOL Dictionary

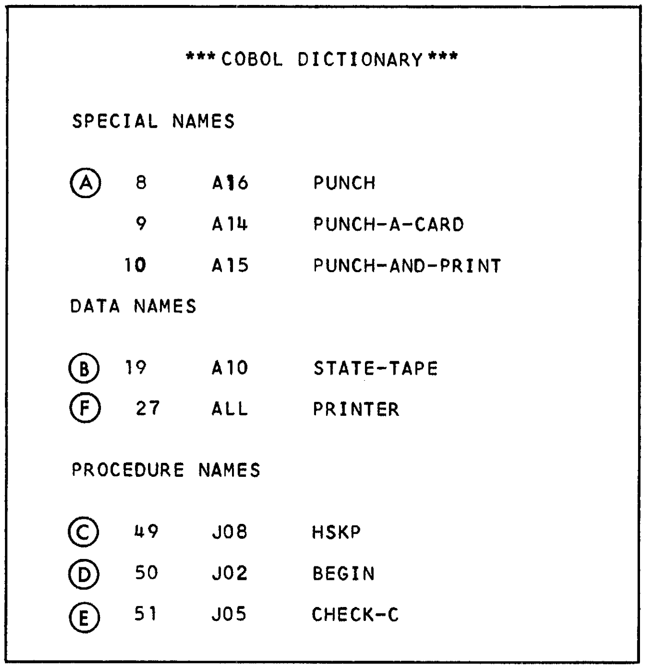
After the COBOL source program and diagnostic messages have been printed, the COBOL dictionary is listed.

The COBOL dictionary is divided into three sections:

Special Names
Data Names
Procedure Names

In each section, all names used in the source program are listed beside the sequence number and the compressed COBOL name developed during compilation by the COBOL processor. This makes it possible to cross-reference items in the COBOL source-program listing and the Autocoder symbolic listing.

Figure 1 shows a sample COBOL dictionary. The line labeled Ⓓ in Figure 1 refers to the procedure name



```
              ***COBOL DICTIONARY***

   SPECIAL NAMES

   Ⓐ   8      A16      PUNCH
       9      A14      PUNCH-A-CARD
      10      A15      PUNCH-AND-PRINT
   DATA NAMES

   Ⓑ  19      A10      STATE-TAPE
   Ⓕ  27      ALL      PRINTER

   PROCEDURE NAMES

   Ⓒ  49      J08      HSKP
   Ⓓ  50      J02      BEGIN
   Ⓔ  51      J05      CHECK-C
```

Figure 1. COBOL Dictionary

BEGIN which appears in the sample COBOL source program shown in Figure 2. The COBOL processor assigned BEGIN the compressed name J02 and assigned the COBOL statement the sequence number 50. In the symbolic listing of the Autocoder expansion (Figure 3), the COBOL processor has used the compressed name J02 as the label for the EQU statement labeled Ⓓ.

To refer to the COBOL source paragraph or statements that produced a particular expansion on the symbolic listing, look at the compressed COBOL name and find its equivalent in the COBOL dictionary. For example, the entry labeled Ⓑ on line 19 of the symbolic listing is used as STATE-TAPE in the COBOL source list. This entry also appears in the data-names section of the COBOL dictionary.

COBOL Diagnostic Usage

Prescan Diagnostics

An optional prescan diagnostic run can be performed by using the 1401 COBOL processor system to check the source program for errors before compilation.

Two types of diagnostic messages can be generated during a prescan run. The first type is name-associated. The name associated messages appear as flags in the dictionary and refer to such errors as COBOL keywords used as data names. The following is an example of name-associated diagnostics:

```
                 TYPE              SOURCE
                   .        .        .
                   .        .        .
                   .        .        .
                 SPEC       M      CARDS
                 SPEC              PRINTER
                 SPEC              OVERFLOW
                 SPEC              LAST-CARD
                 FILE              STATUS-FILE
                  REC       M      CARDS
                   DATA            CARD-ORDER
**KEYWORD          DATA            DIGIT
                   DATA            RECORD
                   .        .        .
                   .        .        .
                   .        .        .
```

In the preceding example, the COBOL keyword, DIGIT, was used as a name, and flagged. In addition, the name, CARDS, was used more than once in the source program and was flagged as being multidefined. If CARDS has not been qualified in all its uses, an error exists.

The second type of prescan diagnostic message describes problems encountered while analyzing the sentence structure of the source program. In the listing, the sequence number of the card containing the error statement relates that which *should have been* coded to that which *was* coded, If, in order to continue analysis of the source program, any part of the source program is dropped, the dropped item(s) are included as part of the diagnostic message. In special cases, the expected part of the diagnostic message might be replaced by a language term or a rule describing a requirement of the COBOL language, such as: ENVIRONMENT DIVISION MUST PRECEDE DATA DIVISION.

To correct program errors of this type, the programmer should analyze the diagnostics in sequence. Also, the correction process can be greatly facilitated if the programmer compares the source program error statement with the corresponding COBOL-prescribed format outlined in the SRL publication *COBL (on Tape) Specifications for IBM 1401,* Form C24-1492. Note, however, that items dropped as a result of an error in sentence structure are not analyzed prior to their deletion.

The following is an example of analyzing COBOL diagnostics:

COBOL SOURCE PROGRAM

```
SEQUENCE                       CARD IMAGE
   .                                .
   .                                .
   .                                .
  30     ENVIRONMENT DIVISION.
  40     CONFIGURATION SECTION.
  50     SOURCE-COMPUTER. IBM-1401.
  60     OBJECT-COMPUTER. IBM-1401 NO-
           OVERLAP
  70     SPECIAL NAMES.
  80       1402-R IS READER
  90       1402-P IS PUNCH
 100       AUTON IS COBOLN
 110     AUTOZ IS COBOLZ
 120       1401-SS A ON STATUS IS LAST-CARD.
 130     DATA DIVISION.
   .                                .
   .                                .
 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 GOBOLN.
 320     PARAGRAPH3.
   .                                .
   .                                .
   .                                .
```

COBOL DIAGNOSTICS

```
SEQUENCE         COMMENTS
   120     AUTOCODER NAME/'.'=1401—SS=
   300     INVALID SUBJECT OR OBJECT=OR=
```

From the preceding example, first consider the sequence number 120. This diagnostic means that in sequence number 120, either an Autocoder name or a period was expected. Instead, 1401-SS was sensed. As shown in the reference-format section of the *COBOL (on tape) Specifications* SRL publication, the special names paragraph is:

```
SPECIAL-NAMES.
-------------
[device-name-1 IS mnemonic-name-1 [device-name-2 IS
               --                                --
     mnemonic-name-2 . . . ]]

[    switch-name-1    [ON STATUS IS condition-name-1]
                       --        --
[OFF STATUS IS condition-name-2]
 ---        --
                               [switch-name-2 . . . ]]

[AUTOCODER-name IS COBOL-name [AUTOCODER . . . ]].
```

The switch-name clause in sequence number 120 (1401-SS etc.) should have preceded the Autocoder-name clauses in sequence numbers 100 and 110.

Secondly, consider sequence number 300. This diagnostic indicates that the error is in sequence number 300. A valid subject or a valid object was expected; OR was sensed. As shown in the *COBOL (on tape) Specifications* SRL publication, the format for relational conditions is defined:

$$\left[\left[\begin{Bmatrix}\textit{data-name}\\ \textit{literal}\\ \textit{arithmetic expression}\end{Bmatrix}\right]\begin{Bmatrix}\text{IS [}\underline{\text{NOT}}\text{] }\underline{\text{GREATER}}\text{ THAN}\\ \text{IS [}\underline{\text{NOT}}\text{] }\underline{\text{LESS}}\text{ THAN}\\ \text{IS [}\underline{\text{NOT}}\text{] }\underline{\text{EQUAL}}\text{ TO}\\ =\end{Bmatrix}\right]$$

$$\begin{Bmatrix}\textit{data-name}\\ \textit{literal}\\ \textit{arithmetic expression}\end{Bmatrix}$$

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, a literal, or an arithmetic expression after LESS for comparison with the implied subject, COBOLZ. Instead, OR was encountered.

The meanings of the symbols 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.
=RAN=	Equal signs bound literal values encountered.
(ALL TO MASTER)	Parentheses bound literal values dropped.
'RUN'/Literal	The slash represents the word OR.

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

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

The sequence number 620 is the number assigned to the card containing the statement in error; UNDECLARED NAME is a language term, bounded by blanks, representing an error that was encountered; = 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)

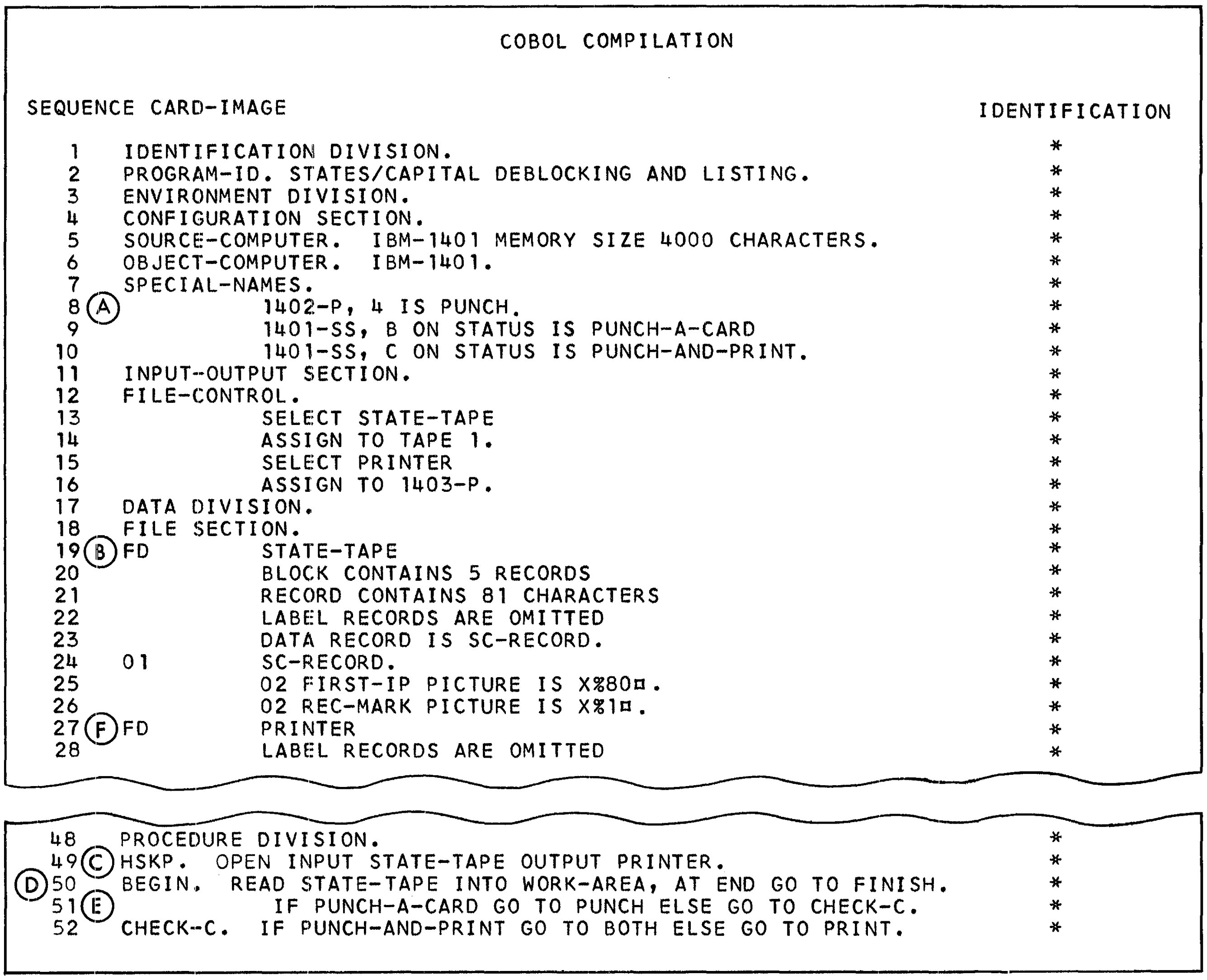
Sequence number 650 is the assigned sequence number of the card containing the statement in error; 'TO' is the literal value, bounded by quotes, expected by the compiler; = ALL = is the literal value, bounded by equal signs, encountered during the run; and (ALL TO MASTER) is the item dropped, bounded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

Cobol Processor Diagnostics

The COBOL processor analyzes each statement for errors as it compiles the COBOL source program. The processor supplies a diagnostic message describing the nature of a detected error and prints the message(s) (ahead of the COBOL dictionary) in this format:

```
                  ***DIAGNOSTICS***

FLAG    REFERENCE    SEQUENCE    MESSAGE    ERROR
 **        B56          14       FORMAT
```



```
                              COBOL COMPILATION

SEQUENCE CARD-IMAGE                                              IDENTIFICATION

    1   IDENTIFICATION DIVISION.                                       *
    2   PROGRAM-ID. STATES/CAPITAL DEBLOCKING AND LISTING.             *
    3   ENVIRONMENT DIVISION.                                          *
    4   CONFIGURATION SECTION.                                         *
    5   SOURCE-COMPUTER.  IBM-1401 MEMORY SIZE 4000 CHARACTERS.        *
    6   OBJECT-COMPUTER.  IBM-1401.                                    *
    7   SPECIAL-NAMES.                                                 *
    8Ⓐ            1402-P, 4 IS PUNCH.                                  *
    9             1401-SS, B ON STATUS IS PUNCH-A-CARD                 *
   10             1401-SS, C ON STATUS IS PUNCH-AND-PRINT.             *
   11   INPUT-OUTPUT SECTION.                                          *
   12   FILE-CONTROL.                                                  *
   13             SELECT STATE-TAPE                                    *
   14             ASSIGN TO TAPE 1.                                    *
   15             SELECT PRINTER                                       *
   16             ASSIGN TO 1403-P.                                    *
   17   DATA DIVISION.                                                 *
   18   FILE SECTION.                                                  *
   19ⒷFD          STATE-TAPE                                           *
   20             BLOCK CONTAINS 5 RECORDS                             *
   21             RECORD CONTAINS 81 CHARACTERS                        *
   22             LABEL RECORDS ARE OMITTED                            *
   23             DATA RECORD IS SC-RECORD.                            *
   24   01        SC-RECORD.                                           *
   25             02 FIRST-IP PICTURE IS X%80¤.                        *
   26             02 REC-MARK PICTURE IS X%1¤.                         *
   27ⒻFD          PRINTER                                              *
   28             LABEL RECORDS ARE OMITTED                            *
```

```
   48   PROCEDURE DIVISION.                                            *
   49ⒸHSKP.  OPEN INPUT STATE-TAPE OUTPUT PRINTER.                     *
 Ⓓ50   BEGIN.  READ STATE-TAPE INTO WORK-AREA, AT END GO TO FINISH.    *
   51Ⓔ             IF PUNCH-A-CARD GO TO PUNCH ELSE GO TO CHECK-C.     *
   52   CHECK-C.  IF PUNCH-AND-PRINT GO TO BOTH ELSE GO TO PRINT.      *
```

Figure 2. COBOL Source Listing

Flag: This indicates the seriousness of the error. If the flag column is blank, the error will not stop compilation. If a double asterisk appears in the flag column, compilation stops and the message COMPILATION SUSPENDED is printed after all statements have been examined.

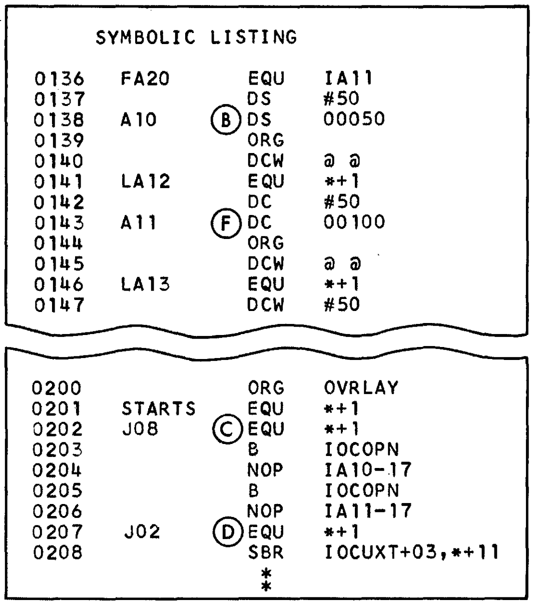
Reference: This shows the code number of the diagnostic message. A detailed explanation of the error can be found by using this code to look up the associated diagnostic in the *Diagnostic Reference List.*

Sequence: This refers to the source program statement in error.

Message: This describes the nature of the error. After all errors have been analyzed and all diagnostic messages have been printed for this phase of compilation, the message TOTAL NUMBER OF ERRORS is printed with a figure indicating the total number of diagnostics that occurred. Then the message COMPILATION SUSPENDED is printed if a double asterisk was printed in the flag column for any of the diagnostic messages given. If none of the diagnostic messages was flagged, the message PRESS START TO CONTINUE is printed, and compilation can be continued.

Following this first list of diagnostic messages, the COBOL dictionary is printed. If a name-oriented error is detected, a diagnostic message will be printed, compilation will be discontinued, and the message RUN DISCONTINUED will be printed.

As compilation continues, discrepancies between the Data Division and the Procedure Division will be detected. If such errors are found, additional diagnostic messages will be printed in this format.



```
      SYMBOLIC LISTING

0136   FA20      EQU    IA11
0137             DS     #50
0138   A10     ⒷDS     00050
0139             ORG
0140             DCW    @ @
0141   LA12      EQU    *+1
0142             DC     #50
0143   A11     ⒻDC     00100
0144             ORG
0145             DCW    @ @
0146   LA13      EQU    *+1
0147             DCW    #50

0200             ORG    OVRLAY
0201   STARTS    EQU    *+1
0202   J08     ⒸEQU    *+1
0203             B      IOCOPN
0204             NOP    IA10-17
0205             B      IOCOPN
0206             NOP    IA11-17
0207   J02     ⒹEQU    *+1
0208             SBR    IOCUXT+03,*+11
                 *
                 *
```

Figure 3. Symbolic Listing

```
FLAGS   REFERENCE  SEQUENCE  COMMENTS      MESSAGE
 **       D14        0113               INVALID EDITING
```

NOTE: Each diagnostic not corrected also appears (altered slightly) as an Autocoder comment line, indicated by an *, in the Autocoder symbolic-program listing.

**: This indicates an error that must be corrected before an object run is attempted.

D14: This is an example of the reference-list code. The *Diagnostic Reference List* contains more detailed explanations of the diagnostic messages.

0113: This is an example of the Autocoder line number of the statement.

*: This is a comments flag to the Autocoder processor.

Message: This is a general description of the error. At the end of compilation, the message NUMBER OF ERRORS NEEDING CORRECTION, followed by the message TOTAL NUMBER OF DIAGNOSTICS, may be printed with figures indicating the number of errors needing correction and the total number of diagnostics. Then this message will be printed to show symbolic listings and/or symbolic punched-card options:

```
IF EXTRA OUTPUT DESIRED
B ON FOR PUNCHED AUTOCODER
D ON FOR PRINTED LISTING
AND PRESS START
IF NO EXTRA OUTPUT DESIRED, PRESS START
```

(Also see *Operating Functions.*)

Diagnostic Reference List

Format Error Messages

These diagnostic messages will be given for format errors that occur in the divisions under which they are listed. In cases where these diagnostic messages appear, the format specified should be carefully reviewed for errors. No reference code is given for these diagnostic messages.

Environment Division Format Errors

1. FORMAT ERROR—ASSIGN
2. FORMAT ERROR—DEVICE NAME 1403-CT
3. FORMAT ERROR—DEVICE NAME 1403-P
4. FORMAT ERROR—SWITCH NAME 1403-P-CB
5. FORMAT ERROR—MEMORY SIZE
6. FORMAT ERROR—SWITCH NAME 1403-P-C9
7. FORMAT ERROR—DEVICE NAME 1402-P
8. FORMAT ERROR—DEVICE NAME 1402-R
9. FORMAT ERROR—SELECT
10. FORMAT ERROR—RESERVE ALTERNATE AREA
11. FORMAT ERROR—SWITCH NAME 1403-SS
12. FORMAT ERROR—SWITCH NAME 1403-P-CV

Data Division Format Errors

1. FORMAT ERROR—FD
2. FORMAT ERROR—RECORDING MODE
3. FORMAT ERROR—BLOCK CONTAINS
4. FORMAT ERROR—RECORD CONTAINS
5. FORMAT ERROR—LABEL RECORD
6. FORMAT ERROR—VALUE OF
7. FORMAT ERROR—DATA RECORD
8. FORMAT ERROR—SIZE
9. FORMAT ERROR—REDEFINES
10. FORMAT ERROR—OCCURS
11. FORMAT ERROR—POINT LOCATION
12. FORMAT ERROR—PICTURE
13. FORMAT ERROR—JUSTIFICATION
14. FORMAT ERROR—LEAVING
15. FORMAT ERROR—VALUE

Procedure Division Format Errors

1. FORMAT ERROR—MOVE
2. FORMAT ERROR—OPEN
3. FORMAT ERROR—MULTIPLY
4. FORMAT ERROR—GO TO
5. FORMAT ERROR—DISPLAY
6. FORMAT ERROR—SUBTRACT
7. FORMAT ERROR—ADDITION
8. FORMAT ERROR—DIVIDE
9. FORMAT ERROR—ALTER
10. FORMAT ERROR—ACCEPT
11. FORMAT ERROR—CLOSE
12. FORMAT ERROR—WRITE
13. FORMAT ERROR—READ
14. FORMAT ERROR—COMPUTE
15. FORMAT ERROR—STOP
16. FORMAT ERROR—PERFORM
17. FORMAT ERROR—IF

Diagnostic Error Messages Inserted during Compilation

Reference Number	*Division*	*Message*	*Comments*
A01	Environment	NEXT CARD OUT OF SEQUENCE	The page and line number combination in the next source program card is not higher in value than that in the preceding card.
A02	Environment Data Procedure	FORMAT ERROR—CONTINUATION OF ALPHANUMERIC LITERALS	If an alphanumeric literal is to be continued, a — symbol must appear in column 7 of the next source program statement line. Also, all lines containing alphanumeric literals must be enclosed in quote symbols. These lines can begin and end in any columns within the range of 12 through 72.
A03	Environment Data Procedure	NUMERIC LITERAL EXCEEDS LEGAL LENGTH	A numeric literal intended for computational use must have from one to 18 numerals. Other numeric literals can contain as many as 120 characters.
A04	Environment Data Procedure	NAME EXCEEDS LEGAL LENGTH	COBOL names may be one to thirty characters long.
A05	Environment Data Procedure	ALPHANUMERIC LITERAL EXCEEDS LEGAL LENGTH	Alphanumeric literals can contain from 1 to 120 characters including blanks and special characters.
A06	Environment Data Procedure	SPECIAL CHARACTER MUST BE AN ALPHANUMERIC LITERAL	Special characters not contained in the COBOL set may only be used in alphanumeric literals. Figure 4 shows the special characters included in the COBOL set.
A07	Procedure	ONLY AUTOCODER CAN BE ENTERED FROM COBOL	There is no provision in COBOL to enter any other language format except Autocoder. This is accomplished by using the ENTER verb.
A08	Identification	PROGRAM-ID NOT SENSED	A PROGRAM-ID paragraph must be present in every COBOL source program.
A09	Identification	IDENTIFICATION DIVISION NOT SENSED	An Identification Division statement must be present in every COBOL source program.
B01	Procedure	PERIOD MUST BE FOLLOWED BY VERB PARAGRAPH OR SECTION NAME	The COBOL statement in reference must be followed by a verb, paragraph, or section name.
B02	Procedure	PERIOD MUST FOLLOW PARAGRAPH OR SECTION	A period must be used to terminate the paragraph or section name in reference.

Card Code	*COBOL*	*1401*	*Meaning*
Blank			space
11	–	–	minus sign hyphen
12	+	&	plus sign
0-1	/	/	division sign
11-4-8	*	*	multiplication sign check protection symbol
12-4-8	)	⌑	right parenthesis
0-4-8	(	%	left parenthesis
0-3-8	,	,	comma
11-3-8	$	$	dollar sign
12-3-8	.	.	period decimal point
3-8	=	#	equal sign
4-8	'	@	quotation mark

Figure 4. Special Characters

Number Reference	*Division*	*Message*	*Comments*
B03	Procedure	KEYWORD DIVISION MUST BE FOLLOWED BY A PERIOD	A period must follow each division name.
B04	Procedure	KEYWORD DIVISION NOT SENSED	The complete name of every division must appear in all source programs.
B21	Environment	CONFIGURATION SECTION NOT SENSED	Each source program must contain a CONFIGURATION SECTION.
B22	Environment	KEYWORD SOURCE-COMPUTER MUST FOLLOW CONFIGURATION SECTION	The first entry in the CONFIGURATION SECTION must be the SOURCE-COMPUTER statement.
B23	Environment	IBM-1401 CLAUSE NOT SENSED	The source program must indicate the types of computers being used for both processing and executing the COBOL program.
B24	Environment	INVALID ENTRY	The statement in reference is not valid in a COBOL source program.
B25	Environment	KEYWORD FILE-CONTROL MUST FOLLOW INPUT-OUTPUT SECTION	The file-description entries must follow the INPUT-OUTPUT SECTION statement.
B26	Environment	PERIOD MUST FOLLOW PARAGRAPH SECTION OR DIVISION	A period must be used to terminate a paragraph, section, or division.
B27	Environment	PERIOD MUST FOLLOW ENTRY	Review this entry using the sequence reference. This entry must be terminated by a period.
B28	Environment	KEYWORD SECTION NOT SENSED	The keyword SECTION must appear in these entries: FILE SECTION, WORKING-STORAGE SECTION, CONSTANT SECTION
B29	Environment	KEYWORD DIVISION NOT SENSED	The complete name of every division must appear in all source programs.
B31	Procedure	MISSING PERIOD AT END OF PROGRAM	The last entry of all source programs must be terminated by a period.
B41	Data	WORKING-STORAGE SECTION OUT OF SEQUENCE	The WORKING-STORAGE SECTION must follow the FILE SECTION in the Data Division.
B42	Data	CONSTANT SECTION OUT OF SEQUENCE	If the source program has a WORKING-STORAGE SECTION, the CONSTANT SECTION must immediately follow it. If the source program does not have a WORKING-STORAGE SECTION, the CONSTANT SECTION must follow the FILE SECTION.
B43	Data	RECORDING MODE MUST BE 1	All 1401 COBOL programs must specify a recording mode of 1.
B44	Data	DATA RECORDS CLAUSE MUST BE IN FD	Each FD must end with a DATA RECORDS clause. This permits the processor to cross-reference the file-description entry and the individual record-description entries of each record.
B45	Data	LABEL RECORDS CLAUSE MUST BE IN FD	This LABEL RECORDS clause must always be present in a FILE DESCRIPTION entry.
B46	Data	DUPLICATE RECORDING MODE CLAUSE	This statement can be removed from the source program.
B47	Data	DUPLICATE BLOCK CONTAINS CLAUSE	This statement can be removed from the source program.
B48	Data	DUPLICATE RECORD CONTAINS CLAUSE	This statement can be removed from the source program.
B49	Data	DUPLICATE VALUE OF CLAUSE	This statement can be removed from the source program.
B51	Data	A PERIOD MUST FOLLOW ENTRY	This statement must be terminated by a period.

Reference Number	Division	Message	Comments
B52	Data	DATA DIVISION NOT FOLLOWED BY VALID SECTION	The Data Division statement must be followed by at least one of these sections: FILE SECTION (contains file- and record-description entries) WORKING-STORAGE SECTION (contains record-description entries) CONSTANT SECTION (contains record-description entries)
B53	Data	DUPLICATE CONSTANT SECTION	A COBOL source program can contain only one CONSTANT SECTION.
B54	Data	DUPLICATE WORKING-STORAGE SECTION	A COBOL source program can contain only one WORKING-STORAGE SECTION.
B55	Data	LEVEL NUMBER MUST BE FOLLOWED BY NAME OR FILLER	All level numbers (01-49, 77 and 78) must be followed by a name or by the word FILLER.
B56	Data	FORMAT ERROR	This statement is written in an invalid format. Error examples: 1. PICTURE IS 999 VALUE IS 10 LEFT JUSTIFIED. The statement must be written as JUSTIFIED LEFT. 2. PICTURE IS 99V X (10). Blanks cannot be used in a picture in this manner.
B57	Data	KEYWORD SECTION NOT SENSED	The word SECTION must appear in these entries: FILE SECTION WORKING-STORAGE SECTION CONSTANT SECTION
B58	Data	KEYWORD DIVISION NOT SENSED	The complete name of every division must appear in all COBOL programs.
B85	Data	FIGCON ALL MUST BE FOLLOWED BY ALPHANUMERIC LITERAL	The ALL figurative constant generates a sequence of characters specified by any non-numeric literal.
B91	Data Procedure	MISSING PERIOD BEFORE PROCEDURE DIVISION	The source program statement that precedes the Procedure Division statement must be terminated by a period.
B92	Procedure	KEYWORD MUST NOT BE USED AS A LABEL	Words listed as COBOL keywords must be used only as specified by the COBOL language.
B97	Data	DATA DIVISION NOT SENSED	A Data Division statement must be included in every source program.
B98	Procedure	PROCEDURE DIVISION NOT SENSED	A Procedure Division statement must be included in every source program.
B99	Environment	ENVIRONMENT DIVISION NOT SENSED	An Environment Division statement must be included in every source program.
C01	Procedure	CONDITION NAMES MUST NOT BE USED AS SUBJECTS	In this example, N1, N2, and N3 are declared as data names with the values defined in the VALUE clauses. 01 NAME1 02 NAME2 88 N1 VALUE IS 1 88 N2 VALUE IS 3, 6 88 N3 VALUE IS 7, 11 Thus, the statement: IF N2 = 4 THEN .... is in error because N2 was previously defined as having a value of 3 or 6.
D01	Data	PICTURE EXCEEDS 30 CHARACTERS	A COBOL PICTURE description can not contain more than 30 characters. For example: The PICTURE description 9(450) consists of six actual characters and is, therefore, properly declared. However, if the programmer had written 450 nines on the coding sheet to describe the same PICTURE, the coding sheet would be in error.

Reference Number	*Division*	*Message*	*Comments*
D02	Data	CONFLICTING EDITING CLAUSES	Only one of the following clauses may be used to edit a field: 1. ZERO SUPPRESS 2. CHECK PROTECT 3. FLOAT DOLLAR SIGN If more than one of these is specified, the last specified clause is chosen for use. Complex editing should be specified by using a PICTURE clause.
D05	Data	CONFLICT BETWEEN PICTURE AND SIZE CLAUSE	If a specified SIZE clause conflicts with a specified PICTURE clause, the PICTURE specifications take precedence over the SIZE specifications. *Example:* SIZE IS 10 conflicts with PICTURE 9(6). The processor will use 6 as the size for the item defined.
D06	Data	CONFLICT BETWEEN PICTURE AND POINT LOCATION CLAUSE	If there is a conflict between a POINT LOCATION and a PICTURE clause, the PICTURE specifications take precedence over the POINT LOCATION specifications. *Example:* PICTURE IS 9V99 conflicts with POINT LEFT 3. The processor will cause the decimal point to be located immediately to the left of the hundreds position of the field. The POINT LOCATION clause specifying that the decimal point be located immediately to the left of the thousands position is ignored.
D07	Data	CONFLICT BETWEEN PICTURE AND CLASS	If there is a conflict between a CLASS and a PICTURE clause, the PICTURE specification takes precedence over the CLASS clause. *Example:* PICTURE IS 999. CLASS IS ALPHANUMERIC In this example CLASS must be specified as NUMERIC because the PICTURE clause specified a numeric field.
D08	Data	CONFLICT BETWEEN PICTURE AND EDITING CLAUSE	If there is a conflict between an editing and a PICTURE clause, the PICTURE specifications take precedence over the editing clause. For example, if the clause: PICTURE IS $$$9 is used and an editing clause specifying FLOAT DOLLAR SIGN is also used, the PICTURE clause takes precedence over the editing clause.
D09	Data	ELEMENTARY ITEM MUST HAVE SIZE OR PICTURE CLAUSE	An element of data (elementary item) must never be subdivided. Each elementary item must be specified by either a SIZE or PICTURE clause in the Data Division.
D11	Data	A RECORD AREA MUST NOT EXCEED 999 CHARACTERS	A record may be described as containing as many as, but no more than, 999 characters.
D12	Data	RLI MUST BE 4 CHARACTERS	The Record Length Indicator (RLI) must be four characters long, including high-order zeros.
D13	Data	RECORD SIZES WITHIN THIS FD MUST NOT CONFLICT	If a FD contains more than one level 01 item, an implied redefinition results in a redefinition of the first 01. If the record size is variable, the DEPENDING clause should be used. This diagnostic message is given because record lengths within the FD were not the same. This condition could cause an error to occur at object program execution time.
D14	Data	INVALID EDITING	The editing is not specified according to the COBOL requirements. All editing for the entry in reference was omitted by the processor. Examples of such illegal coding are: 1. Invalid symbols in a PICTURE clause. 2. The size of the item to be edited exceeds 120 characters. 3. Zero suppression is specified for only the high-order position of a field. 4. DB (debit symbol) or CR (credit symbol) is not specified for the rightmost or leftmost positions of the field.
D99	Data	PICTURE CLAUSE INVALID WITH GROUP ITEM	Only elementary items can have a PICTURE clause.

Reference Number	*Division*	*Message*	*Comments*
E01	Data (IOCS)	HARDWARE DEVICE MULTI-DEFINED	IBM 1401 Data Processing System devices can not be multi-defined. The example shown illustrates two combinations that will cause multiple-definition errors. Statements 1 and 2 or statements 1 and 3 cause a multiple definition of the 1402 card punch because 1402-P is assigned to *punch* in SPECIAL NAMES. Statements 2 and 3 excluding statement 1 will cause a multiple-definition error because two files cannot be assigned to the same device. *Example:* 1. SPECIAL NAMES. 1402-P, 4 IS PUNCH. INPUT-OUTPUT SECTION. FILE-CONTROL. 2. SELECT FILE2 ASSIGN TO 1402-P. 3. SELECT FILE3 ASSIGN TO 1402-P.
E02	Data (IOCS)	BLOCK SIZE MUST NOT EXCEED OBJECT-COMPUTER SIZE	Blocking specified in the COBOL source program must not specify more than the core-storage capacity of the object computer.
E03	Data (IOCS)	FORM 3 RECORDS NOT PERMITTED	COBOL does not permit unblocked, variable-length records.
E04	Data (IOCS)	BLANKS IN HEADER-LABEL ID NOT PERMITTED	Blanks in header-label ID will be replaced with a period.
E05	Data (IOCS)	UNIT RECORD FILES MUST HAVE STANDARD RECORD SIZES	These sizes must be specified for unit-record files: Card Reader—80 characters Card Punch—80 characters Printer—100 or 132 characters
E06	Data (IOCS)	FORM 4 RECORDS-BLOCK SIZE MUST BE IN CHARACTERS	The block size for Form-4 records must not be specified by using the keyword RECORD(S). Thus, the example shown is in error. *Example:* BLOCK CONTAINS 400 RECORDS
E07	Data (IOCS)	FILE MUST BE SELECTED	Each file to be processed by the object program must be named in a SELECT file-name entry. The name must be unique within the source program.
E08	Data (IOCS)	BLOCK SIZE MUST NOT EXCEED 4 DIGITS	The block size specification can contain a maximum of four digits, including high-order zeros.
E09	Data (IOCS)	RECORD SIZE MUST NOT EXCEED BLOCK SIZE	A record-size specification must designate more characters than the size specification for the block that contains the record.
E10	Data (IOCS)	UNIT RECORD FILES MUST HAVE FORM 1 RECORDS	Blocked records are not permitted in unit-record files.
E11	Data (IOCS)	UNIT RECORD FILE-LABEL RECORDS MUST BE OMITTED	The LABEL RECORDS clause for unit-record files must be specified as OMITTED.
E12	Data (IOCS)	FILE MUST BE OPENED	The OPEN verb is used to indicate the processing of one or more input and/or output files. At least one of the two clauses must be written. The clauses may specify that one or more files be opened.
E13	Data (IOCS)	UNIT RECORD FILES MUST NOT RESERVE ALTERNATE AREAS	All alternate areas needed for unit-record files are reserved automatically by the processor. Thus, the programmer must not reserve alternate areas for unit record files. *Error example:* SELECT READER ASSIGN TO 1402-R RESERVE 1 ALTERNATE AREA.
F01	Procedure	END STATEMENTS NOT ASSOCIATED WITH THE FOLLOWING FILES	At least one implicit or explicit AT END statement must be 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.

Reference Number	Division	Message	Comments
H01	Procedure	INVALID USE OF SUBSCRIPTING	This diagnostic message occurs when invalid subscripting is used without an OCCURS clause. *Error example:* DATA DIVISION. . . . FILE SECTION. 01 NAME1 02 NAME2† 03 NAME3 PROCEDURE DIVISION. . . . MOVE NAME2 (3) TO WORK AREA †The procedure statement specifies subscripting. Therefore, NAME2 must contain an OCCURS clause.
I00	Procedure	A-FIELD EXCEEDS B-FIELD	The field being moved is larger than the field to which it is to be moved. High-order positions will be dropped.
I01	Procedure	NON-NUMERIC FIELD USED IN COMPUTATION	The referenced field is arithmetic and not defined as numeric.
I02	Procedure	EDIT MASK TOO SMALL	A GIVING, COMPUTE, or MOVE using edited fields too small to accept the field being placed into the edited mask will produce this diagnostic message. *Example:* Moving field 12345 to an edited mask $$$99 is erroneous because this edit mask will accept only four characters.
I03	Procedure	A-FIELD EXCEEDS 18 DIGITS	Fields used in arithmetic computations and expressions must not exceed 18 digits.
I05	Procedure	B-FIELD EXCEEDS 18 DIGITS	See *I03*.
I06	Procedure	NON-NUMERIC FIGCON USED IN COMPUTATION	Only the figurative constant ZERO can be used in arithmetic computations.
I07	Procedure	INVALID SOURCE	If a source-program statement is determined to be meaningless, this diagnostic message is given. Following this message will be an expansion of the STOP RUN literal.
I08	Procedure	INTERMEDIATE RESULTS MUST NOT EXCEED 20 DIGITS	Fields requiring decimal alignment in arithmetic computation must not have intermediate results that exceed twenty digits.
I09	Procedure	ALPHANUMERIC TO NUMERIC MOVE	This diagnostic message is given if a group item (declared as numeric) is moved to an elementary numeric field. All group items are classed as alphanumeric.
I10	Procedure	INVALID SOURCE—IN PRECEDING STATEMENT	If an incorrect source program statement is detected in a DISPLAY statement, this error message is given. *Example:* DISPLAY ON (name) is incorrect, but DISPLAY UPON (name) is the correct format.
I11	Procedure	ROUNDED EXCLUDES LEFT JUSTIFY CLAUSE	If left justification is used in conjunction with the ROUNDED option, justification is ignored by the processor during compilation.
I12	Procedure	GROUP ITEMS MUST NOT HAVE IMPLIED DECIMAL	Decimal alignment is not performed when group items, classified as numeric, are moved.
I13	Procedure	GROUP ITEM MUST NOT BE USED IN COMPUTATION	Only elementary items can be used in arithmetic computations.

Reference Number	*Division*	*Message*	*Comments*
I15	Procedure	INVALID USE OF GROUP	1. When either of the conditionals IF POSITIVE or IF NEGATIVE is used, the field referred to must be elementary. 2. When either the conditional IF NUMERIC or IF ALPHABETIC is used, the field referred to should be elementary. If the fields are not elementary, the diagnostic message will be given to remind the programmer that he should review the units positions of fields being tested for zone bits.
I17	Procedure	ITEM EXCEEDS 20 DIGITS	A numeric field defined to be used for computational purposes must not be larger than twenty digits. If such a numeric field exceeding twenty digits is compared, this diagnostic message is given. An instruction to perform a non-numeric compare is also given.
I18	Procedure	CLASS CONTRADICTION	This diagnostic message indicates that the fields being operated on should be reviewed to determine if there is a contradiction in classes. For example, a numeric field compared to a non-numeric figurative constant would cause this diagnostic message to appear.
I19	Procedure	INVALID USE OF EDITING	Editing must be performed on data fields that appear in procedure statements with the appropriate verbs. *Example:* 1. The field referred to by the ACCEPT verb must not have editing. 2. Editing must not be specified for an integer referred to by name in the TIMES option of the PERFORM verb.
I20	Procedure	SIZE OF LITERAL MUST EQUAL 1	If the literal must be one character long (for example, in use with the EXAMINE verb), this message is given.
I21	Procedure	INVALID SUBSCRIPTING	*Error example:* DATA DIVISION. 01 TABLE 02 NAME1 OCCURS 3 TIMES SIZE IS 2 PROCEDURE DIVISION. MOVE NAME1 (1,2).......... The statement in the Procedure Division contradicts the declaration for NAME1.

COBOL Tape-Print Program

Conditions may arise that the COBOL processor recognizes as being instrumental in causing a failure. In these cases, the message SYSTEM FAILURE is printed on the printer. The processor then calls in a storage- and tape-print program. The contents of core storage and the tape work files are printed on the printer.

Among the conditions that may cause a system failure are:

1. An end-of-file indication was sensed by the processor before the end of the file was reached.
2. An excessive number of tape read or tape write errors occurred on the work tape.
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.

COBOL Phase Descriptions

The IBM 1401 COBOL processor has ten sections. Each section performs a particular function in converting the COBOL source-program statements to Autocoder statements.

Phase A01

1. Reads the COBOL source program from cards or tape.
2. Assigns a sequence number to each entry.
3. Lists the contents of each source-program card on the printer.

Phase A02

Processes the Identification Division.

Phase A0B

Conditions storage and reads in phase A03.

Phase A03

1. Conditions words and operators for further processing.
2. Checks the source program for errors in the use of the character set, names, picture clauses, and word lengths.
3. Checks the program for the ENTER verb and conditions the included Autocoder statements for further processing.

Phase A04

Searches for the various divisions of the source program and calls in phases A05, A06, and A07 to compress keywords and delete unnecessary words as follows:

Phase A05	Environment Division keywords
Phase A06	Data Division keywords
Phase A07	Procedure Division keywords.

Phase A20

This phase is called conditionally to place Autocoder entries on tape 4 for future merging by the J phases.

Phase BD1

Diagnoses the Environment Division.

Phase BD2

Checks the FILE SECTION for syntax errors and checks for duplicate entries in the Data Division (first pass).

Phase BD3

Checks the Procedure Division verbs for syntax errors (first pass).

Phase BD4

Checks the CONSTANT and WORKING-STORAGE sections of the Data Division for syntax errors (second pass).

Phase BD5

Checks the Procedure Division verbs for syntax errors (second pass).

Phase BD6

1. Calls phases BT1, BT2, and BT3 for printing diagnostic messages.

Phase BT1	Environment Division diagnostic table
Phase BT2	Data Division diagnostic table
Phase BT3	Procedure Division diagnostic table.

2. Prints all diagnostic messages and determines whether compilation is to be suspended.

Phase B01

1. Conditions the source program for further processing.
2. Converts level numbers of data description entries to internal level operators.

Phase B02

1. Builds a table of Procedure Division literals and level 88 VALUES to be processed by phase B11. Duplicate literals are eliminated.
2. Builds a table of Data Division literals and eliminates duplicates.

Phase B11

1. Calculate class, size, and decimal count for data and procedure literals.
2. Writes literal declarations on tape 4.

Phase B03

1. Builds a table of procedure names used in the COBOL source program.
2. If the program has an ENTER verb, the COBOL names referred to by Autocoder statements following the ENTER verb are entered into the same table as the procedure names.

Phase B04

Uses the tables built by phase B03 to convert procedure names to reference names for further processing.

Phase B05

1. Builds a table of names used in procedural statements, such as REDEFINES and DEPENDING, and COBOL names referred to by Autocoder statements following the ENTER verb.
2. Compresses data names, used in procedure statements, for further processing.

Phase B06

Matches COBOL source programs data names and special names with the processor-produced compressed names and generates a unique compressed name for each name used in the COBOL program.

Phase B07

Builds a table of processor-produced compressed names used in qualification. It also builds a table of declaration names and procedural names used in qualification.

Phase B08

Develops a unique compressed name from the qualified names used in procedure statements.

Phase B09

1. Lists all COBOL source names according to Environment Division special names, data names, and procedural names categories. Generated sequence numbers are listed with these names.
2. If the source program contained errors in names, the errors are detected and phase B10 is called.

Phase B10

1. Analyzes errors detected during the execution of phases B04, B06, and B08.
2. Source names which have multiple definitions or are undefined are listed on the printer.

Phase C01

1. Phase C01 is executed if a SPECIAL-NAMES paragraph appears in the source program.
2. Generates statements to equate Autocoder names to COBOL names.

Phase C02

Processes level 88 items.

Phase C03

Processes data values if level 88's are not used in the source program.

Phase D01

1. Analyzes PICTURE clauses to determine size, class, decimal count, and editing.
2. Builds a table of all level 01 items for further processing.

Phase D02

1. Builds a description for each Data Division entry. The information in the source declarations is used as a basis for building this table.
2. Updates these descriptions with information obtained from the PICTURE analysis.
3. Generates diagnostic messages if discrepancies are detected.

Phase D03

1. Phase D03 is called if editing is used.
2. Builds and declares edit masks and completes the edit section of the data description. Writes editing masks on tape 4.

Phase D04

1. Establishes the group item size by totaling the sizes of elementary items.
2. Provides padding for OCCURS clauses.
3. Controls redefinition.
4. Generates origins, storage declarations, and equates.
5. Processes subscript levels.

Phase D05

1. Controls the placement of word marks for each entry.
2. Processes file record sizes and equates.
3. Processes nested redefinition entries.

Phase D06

1. Separates data and places delimited items from D05 on the proper output tape.
2. Expands macros and writes storage declarations on tape 4.

Phase E01

Produces the JOB card for the object program.

Phase E02

1. Examines the source program for control and IOCS information.
2. Stores this data, in an abbreviated form, in core storage.
3. If no files are declared, phase E04 is called.

Phase E03

1. Executes any required internal computations, based upon data stored by E02.
2. Completes DIOCS-DTF information by examining all of the data obtained from the source program and selecting the appropriate implied specifications.

Phase E04

1. Analyzes the legality of the source data.
2. Selects the appropriate set of diagnostic messages.

Phase E05

1. If no files were defined:
   a. Writes CTL and ORG statements on tape unit 4.
   b. Writes diagnostic message indicators on tape unit 4, if appropriate.
   c. Calls phase F01.
2. Writes CTL and DIOCS statements on tape unit 4.
3. Writes diagnostic message indicators, if appropriate.
4. Writes DIOCS entries on tape unit 4.

Phase E06

Writes DA entries on tape unit 4, if appropriate.

Phase E07

Writes DTF entries and appropriate diagnostic message indicators on tape unit 4.

Phase F01

Processes these portions of the Procedure Division:
a. OPEN—Deletes INPUT and OUTPUT designations.
b. CLOSE—Deletes rewind options.
c. WRITE—Converts WRITE to WRITE FROM if such form is appropriate.

Phase F02

1. Processes conditional expressions.
2. Processes COMPUTE statements.

Phase F03

1. Processes conditional expressions.
2. Processes READ statements.

Phase F04

1. This phase is called if the PERFORM verb is used in the Procedure Division.
2. Generates labels for names associated with the PERFORM statements and inserts them into the symbolic entries developed from these PERFORM statements.
3. Associates the last name in PERFORM statement with the label and builds a table of these elements.

Phase F05

1. Searches for procedure names and inserts any necessary labels such as RETURN-LINKAGES.
2. Searches the table built in phase F03 for the procedure names found, and conditions labels for further processing.

Phase G01

1. Breaks down the following COBOL source statements into macro form:
   a. ADD
   b. SUBTRACT
   c. MULTIPLY
   d. DIVIDE
   e. MOVE
   f. PERFORM (options 1 and 2)
   g. READ (no AT END)
2. Conditions PERFORM (options 3, 4, and 5) and READ (AT END) for further processing.

Phase G02

1. Processes arithmetic operators in arithmetic expressions and COMPUTE statements.
2. Processes relational operators in IF and UNTIL expressions.
3. Passes generated labels and logic connectors to phase G03.

Phase G03

1. Processes conditional sentences.
2. Ties in generated labels for READ linkage, UNTIL, and IF with conditional sentences.

Phase G04

1. Builds macro statements.
2. Creates linkage macros for PERFORM options 3, 4, and 5.
3. Determines the number of temporary object-machine storage areas needed for arithmetic computations.

Phase G05

Eliminates redundant linkages in relational statements.

Phase G06

Eliminates redundant, generated, temporary, object-machine storage areas. This fully optimizes arithmetic statements.

Phase H01

1. Collects data, device, switch, and literal descriptions and converts them to table entries.

2. Processes subscript macros.
3. Inserts the appropriate data name in the symbolic statements after each occurrence of a subscript data name.

Phase H02

Prepares storage for phases H03, H04, and H05.

Phase H03

1. Initializes the table of data descriptions.
2. Clears storage.
3. Calls phase H04.

Phase H04

1. Builds a table of data descriptions in storage. Entries that have already been processed are deleted.
2. Calls phase H05 when the Procedure Division is recognized.

Phase H05

1. Merges data descriptions after each name in the Procedure Division that has a description.
2. Returns to phase H02 if necessary.

Phase I01

1. Selects the appropriate model statements for fixed expansions, for the ADVANCING option, for STOP RUN, and switches.
2. Conditions relational macros for further processing.

Phase I02

1. Establishes parameters for DISPLAY, ACCEPT, and GO TO DEPENDING routines.
2. Expands IF NUMERIC and IF ALPHABETIC macros.
3. Selects appropriate model statements for the STOP *literal* statement.
4. Makes a diagnostic scan of arithmetic expressions.

Phase I03

1. Conditions input data for further processing and scans data for validity.
2. Conditions GIVING, POSITIVE, NEGATIVE, and relational macros for further processing.
3. Makes a diagnostic scan of arithmetic expressions.

Phase I04

1. Selects appropriate model statements and subroutines for MOVE and MOVE ALL macros.
2. Conditions POSITIVE and NEGATIVE macros for further processing.

Phase I05

Conditions arithmetic macros for further processing by I06.

a. Keeps a record of intermediate accumulators.
b. Calculates decimal alignment.
c. Optimizes model statements to be selected.

Phase I06

1. Selects appropriate model statements for arithmetic macros.
2. Conditions relational macros for further processing.

Phase I07

1. Selects appropriate model statements for all arithmetic macros that have not been processed up to this phase.
2. Selects appropriate model statements for the POSITIVE and NEGATIVE macros.
3. Makes a diagnostic scan of GIVING and GIVING (ROUNDED) macros.
4. Processes rounding, editing, and decimal alignment of GIVING macros.
5. Selects appropriate model statements for GIVING macros.

Phase I08

1. Determines the type of comparisons to be set up for relational statements.
2. Calculates decimal alignment for these comparisons if it is necessary.
3. Selects appropriate model statements and subroutines.
4. Makes a diagnostic scan of relational macros.

Phase I09

1. If no IOCS is specified, J1 is called.
2. Translates DIOCS and DTF entries, read from tape unit 4, into an internal macro format.

Phase SP1

Copies the remainder of the GETX tape into the PUTX tape.

Phase I10

1. Outputs call for Autocoder statements by expanding the DIOCS and DTF macros.
2. Controls the manipulation of the DIOCS and DTF logic tables.

Phases S01 through S05

Are DIOCS logic tables.

Phases T02 through T09

Are DTF logic tables.

Phase SP2

Copies the remainder of the GETX tape onto the PUTX tape.

Phase J01

1. Substitutes parameters into model statements selected from the library.
2. Provides for reiteration if all of the model statement tables will not fit into core storage.
3. If no IOCS is specified, the DI and DT tables are not retrieved.

Phases D11 through D19

Are DIOCS Autocoder model statement tables.

Phases DT1 through DT0

Are DTF Autocoder model statement tables.

Phases JE2 through JE6

Are COBOL Autocoder model statement tables.

Phase SP3

Copies the remainder of the GETX tape onto the PUTX tape.

Phase JA7

Conditions operation codes and substitutes constants into the Autocoder instructions.

Phase J07

1. Sets up routing linkages.
2. Calculates subscript addresses.
3. Generates INCLD for subroutines.

Phase J08

Selects from tape 4 the items listed and writes them on PUT TAPE in this sequence:

1. JOB card
2. CTL (control) card
3. ORG (origin) card
4. DIOCS entries
5. DTF entries
6. Procedure literals
7. Edit masks
8. Storage declarations
9. File areas
10. Generated INCLD
11. Generated constants, index registers, temporary storage accumulators, and tally registers
12. Data literals and data moves
13. Procedure instruction from GET TAPE.

Phase J09

1. Prints diagnostic messages that occurred because conflicts exist between the entries in the Data Division and the Procedure Division.
2. Merges Autocoder symbolic statements with diagnostic messages.
3. Calls in phases JT1-JT3.

Phases JT1 through JT3

Contain IOCS data and procedure diagnostic tables.

Phase J10

1. Writes Autocoder statements on tape unit 4 for assembly of the machine-language object program.
2. Merges the COBOL source program with the Autocoder statements if the source symbolic listing option is exercised.

COBOL Subroutines

The subroutines used by the 1401 COBOL processor are described here.

The subroutine chart outlines each subroutine and gives the subroutine name with the reason the subroutine is called by the processor.

Subroutine (Mnemonic)	*Subroutine*	*Reason Subroutine Is Called*
XMN	Examine	Use of the EXAMINE verb.
XXJ	Subscript-1	Use of one-level subscripting.
XXK	Subscript-2	Use of two-level subscripting.
XXL	Subscript-3	Use of three-level subscripting.
YAQ	Alpha compare	Alphabetic record with subfields compared to any data item.
YCL	Figcon compare	Field whose size is greater than 1 being compared to a figurative constant whose size is greater than 1.
YIN	If numeric	Any alphanumeric field whose size is greater than 1 being tested for a numeric value.
YIP	If alphabetic	Any alphanumeric field whose size is greater than 1 being tested for an alphabetic value.
ZAX	Accept	Use of the ACCEPT verb.
ZDY	Display	Use of the DISPLAY verb.
ZET	Editing	Use of: 1. COBOL zeros 2. Floating + or — 3. DB or single +
ZFZ	Exponentiation-1	Raising an expression by an integer exponent.
ZGP	Go to depending	Use of DEPENDING option of GO TO statement.
ZML	Move all	Use of ALL option of MOVE verb.
ZMR	Move record	Use of a record with subfields in a MOVE statement, except in the case when the other field is a record of equal length.
ZZZ	Basic package	Used with any subroutine.
ZXZ	Exponentiation-2	Raising an expression by a non-integer exponent.

Halts and Messages

Compile-Time Halts

Contents of I-Address Register	*Phase*	*Message*	*Explanation*	*Operator Action*
056	Program Boot	None	Tape transmission error occurred while trying to load the system.	Rewind the system tape on unit 1 and press the Tape Load key.
155	Any	None	Sense switch B should not be on during compilation of a COBOL program.	Turn off sense switch B and press the start key to continue.
698	Any	None	An error has occurred in the GETX subroutine.	Do a manual branch to 359 to obtain storage and tape print.

Contents of I-Address Register	*Phase*	*Message*	*Explanation*	*Operator Action*
808	UPDAT	None	An excessive number of read errors on tape unit n.	1. Press start key to try to reread the record, or 2. Change the tape and recompile.
842	UPDAT (Systems Run)	INCORRECT UPDATE CARD, OR PHASE NOT FOUND	A deck of UPDAT cards is in error.	Correct deck of UPDAT cards.
842	UPDAT (Systems Run)	END OF TAPE ON UNIT 2	End-of-reel occurred on tape unit 2 during system run.	Use larger reel of tape. Restart system run.
892	Any	TE n	A read error has occurred on tape unit n. This halt indicates ten failures to read correctly.	1. Press the start key to try to reread the record, or 2. Change the tape and recompile.
1616	BD4	None	End of program sensed too soon.	Do a manual branch to 359 to get storage and tape prints.
1663	Tape-Print Program	READ ERROR-PRESS START TO REREAD OR RESET START TO SKIP RECORD	(Self-explanatory.)	If the record cannot be skipped, branch manually to 1729 to get the next tape file.
2031	BD6	COMPILATION SUSPENDED	Invalid source program statements have been detected.	Correct the source deck as indicated by the diagnostic messages and recompile the program.
2094	Control Program	NO CONTROL CARD-RELOAD CARDS AND PRESS START	The first cards of a run must be: 1. COBOL RUN (for compilations), and 2. SYSTEMS RUN (for updating or copying the system).	Insert the correct control card and press the start key to continue.
2081	BD6	PRESS START TO CONTINUE	Source program statements require diagnostic messages.	Press the start key to continue.
2149	Control Program	None	The I-O check-stop switch is off, but a card-read error has been detected.	Run out the cards, turn on the I-O check-stop switch and restart.
2221	Control Program	CANT READ UNIT 1	Systems tape cannot be read.	Press the start key to reread. Re-extract the system tape from the PID tape.
2669	J10	END OF REEL	An end-of-reel condition was detected while Autocoder statements were being written on tape 4.	Replace the reel on tape unit 4 with a reel containing more tape footage and restart.
2821	Control Program	None	An end-of-reel condition was detected during the systems run.	Replace the reel on tape unit 2 with a reel containing more footage and restart the run.
2757	B10	RUN DISCONTINUED	Source-program statement error caused compilation to stop.	Correct the source program and restart compilation.
2767	J10	None	This halt permits the user to exercise one of the extra output options at the end of job.	Arrange the sense switches and press the start key.
2818	J10	END OF COMPILATION—AUTOCODER ON TAPE UNIT 4	Final halt at end of job.	
3262	J07	None	End of program is sensed too soon (processor error).	Do a manual branch to location 359 for storage and tape prints.
3486	SP 1 / SP 2 / SP 3	None	End of reel condition on tape 5 or 6.	Remove the reel on the affected tape unit. Replace the reel with another containing more tape footage.

Assembly-Time Halts

For halts that can occur during the assembly of the machine-language object program, see *Autocoder (on Tape) Language Specifications and Operating Procedures, IBM 1401 and 1460* (C24-3319).

Object-Time Halts

For halts that can occur during execution of the object program, see *Input/Output Control System for IBM 1401* (C24-1462).

Two additional error messages can occur. These are associated with error halts in the Z X Z subroutines:

Message	*Explanation*
MINUS VALUE	Negative bases cannot be raised to other than an integer power.
VALUE TOO LARGE	The maximum length of a value used for computation is 18 digits.

If the start key is pressed when either of these halts occurs, the answer developed from computation will be zero and the program will continue.

Sample Program: Table of Salaries

This program computes a set of values, arranges them in a table and prints them.

A monthly salary of $500 is used to compute the corresponding weekly and annual salaries, and the resulting figures are printed in columns titled *Weekly, Monthly,* and *Annual.* The monthly salary figure is then increased by $10 and the procedure is repeated. This process is continued until the table shows the corresponding figures for all monthly salaries from $500 to $1000 at increments of $10.

The format for the printed table is:

Print positions	*Heading*
1-46	blank
47-52	WEEKLY
53-55	blank
56-62	MONTHLY
63-65	blank
66-71	ANNUAL
72-132	blank

The corresponding figures are printed below the headings.

To run the object program after it has been compiled and assembled:

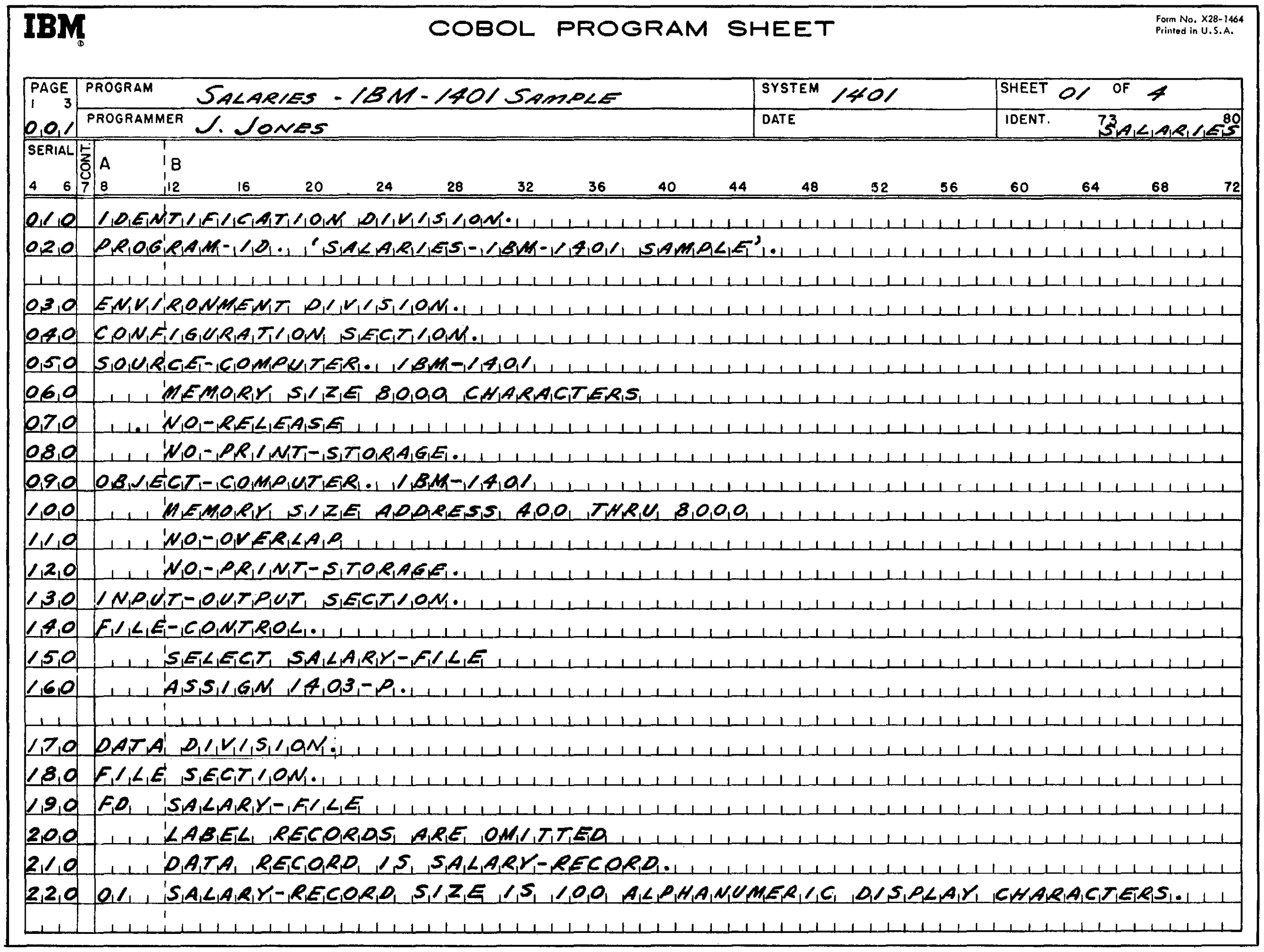
1. Place the assembled deck for the Table of Salaries program in the card reader. This deck has 70003 punched in columns 76-80.
2. Ready the printer.
3. Turn the I/O check-stop switch on.
4. Press the load key on the card read-punch.
5. Press the start key.

The salary table will be printed followed by the message TABLE VALUES ARE CORRECT if all the calculations are correct. If the totals do not balance, the message TABLE VALUES ARE NOT CORRECT will be printed.

The SOURCE-COMPUTER and OBJECT-COMPUTER entries for this program should be changed, if necessary, to indicate the storage size of the 1401 used to compile and run this sample program.

Figure 5 is the source program as it is written on the COBOL program sheet.

Figure 6 shows the actual Table of Salaries printed during the running of the object program.



IBM COBOL PROGRAM SHEET

Form No. X28-1464
Printed in U.S.A.

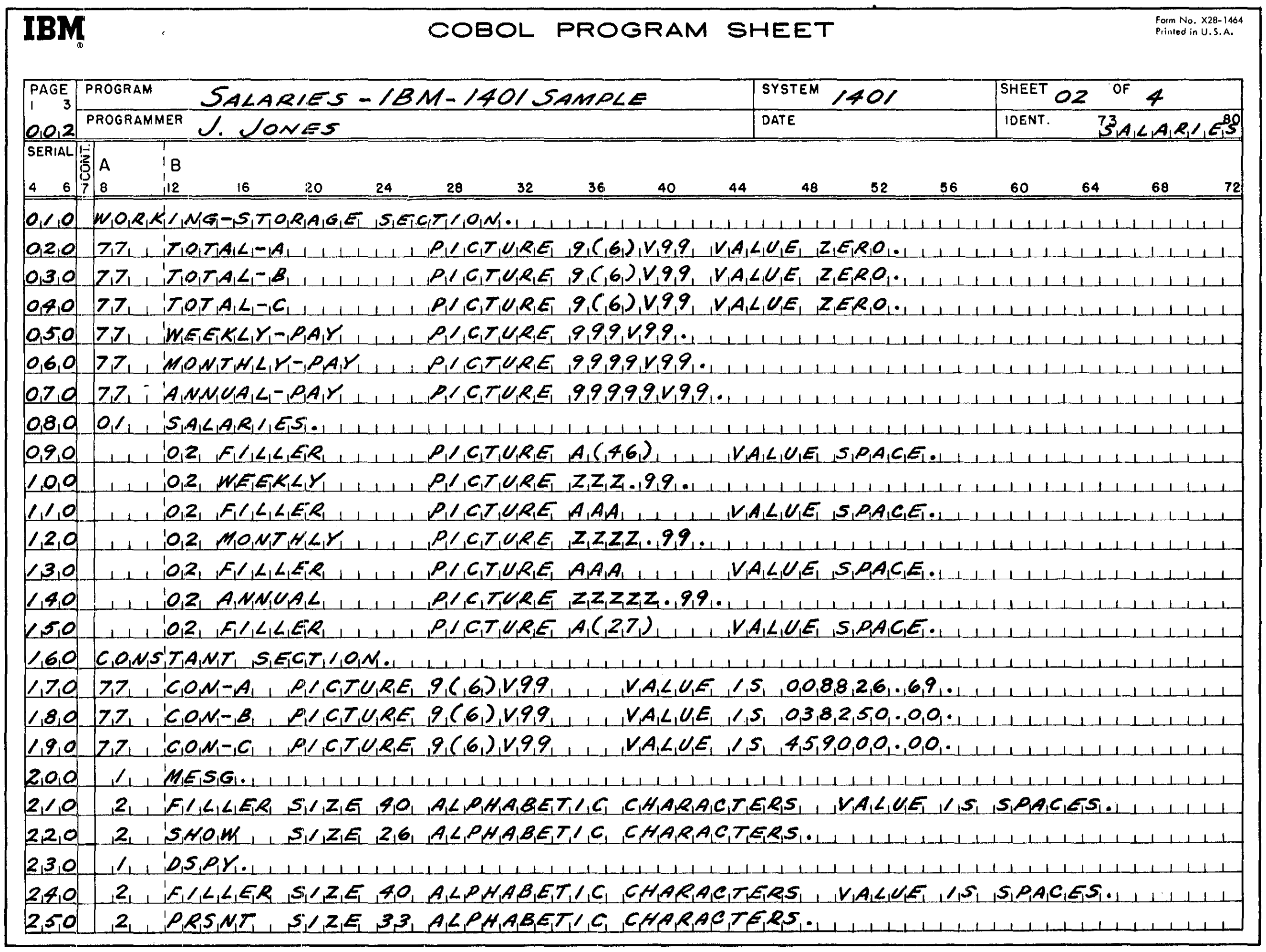
PAGE 001	PROGRAM SALARIES - IBM-1401 SAMPLE	SYSTEM 1401	SHEET 01 OF 4
	PROGRAMMER J. JONES	DATE	IDENT. SALARIES

```
010 IDENTIFICATION DIVISION.
020 PROGRAM-ID. 'SALARIES-IBM-1401 SAMPLE'.

030 ENVIRONMENT DIVISION.
040 CONFIGURATION SECTION.
050 SOURCE-COMPUTER. IBM-1401
060     MEMORY SIZE 8000 CHARACTERS
070     NO-RELEASE
080     NO-PRINT-STORAGE.
090 OBJECT-COMPUTER. IBM-1401
100     MEMORY SIZE ADDRESS 400 THRU 8000
110     NO-OVERLAP
120     NO-PRINT-STORAGE.
130 INPUT-OUTPUT SECTION.
140 FILE-CONTROL.
150     SELECT SALARY-FILE
160     ASSIGN 1403-P.

170 DATA DIVISION.
180 FILE SECTION.
190 FD  SALARY-FILE
200     LABEL RECORDS ARE OMITTED
210     DATA RECORD IS SALARY-RECORD.
220 01  SALARY-RECORD SIZE IS 100 ALPHANUMERIC DISPLAY CHARACTERS.
```

Figure 5. COBOL Sample (Part 1 of 4)



IBM COBOL PROGRAM SHEET

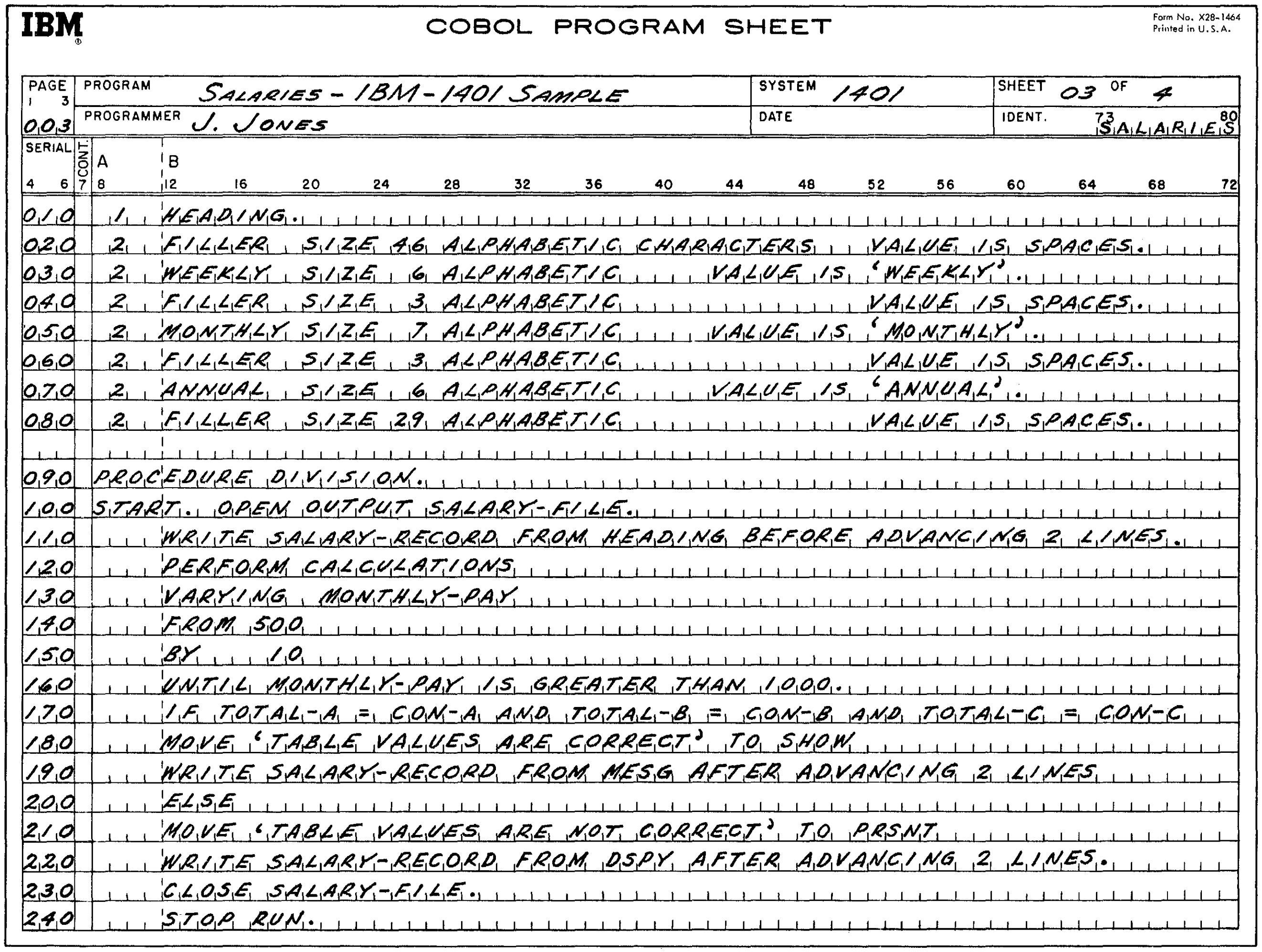
Form No. X28-1464
Printed in U.S.A.

PAGE 002	PROGRAM SALARIES - IBM-1401 SAMPLE	SYSTEM 1401	SHEET 02 OF 4
	PROGRAMMER J. JONES	DATE	IDENT. SALARIES

```
SERIAL  A   B
4  6  7 8   12  16  20  24  28  32  36  40  44  48  52  56  60  64  68  72

010    WORKING-STORAGE SECTION.
020    77  TOTAL-A          PICTURE 9(6)V99  VALUE ZERO.
030    77  TOTAL-B          PICTURE 9(6)V99  VALUE ZERO.
040    77  TOTAL-C          PICTURE 9(6)V99  VALUE ZERO.
050    77  WEEKLY-PAY       PICTURE 999V99.
060    77  MONTHLY-PAY      PICTURE 9999V99.
070    77  ANNUAL-PAY       PICTURE 99999V99.
080    01  SALARIES.
090        02 FILLER        PICTURE A(46)    VALUE SPACE.
100        02 WEEKLY        PICTURE ZZZ.99.
110        02 FILLER        PICTURE AAA      VALUE SPACE.
120        02 MONTHLY       PICTURE ZZZZ.99.
130        02 FILLER        PICTURE AAA      VALUE SPACE.
140        02 ANNUAL        PICTURE ZZZZZ.99.
150        02 FILLER        PICTURE A(27)    VALUE SPACE.
160    CONSTANT SECTION.
170    77  CON-A  PICTURE 9(6)V99     VALUE IS 008826.69.
180    77  CON-B  PICTURE 9(6)V99     VALUE IS 038250.00.
190    77  CON-C  PICTURE 9(6)V99     VALUE IS 459000.00.
200     1  MESG.
210     2  FILLER SIZE 40 ALPHABETIC CHARACTERS  VALUE IS SPACES.
220     2  SHOW   SIZE 26 ALPHABETIC CHARACTERS.
230     1  DSPY.
240     2  FILLER SIZE 40 ALPHABETIC CHARACTERS  VALUE IS SPACES.
250     2  PRSNT  SIZE 33 ALPHABETIC CHARACTERS.
```

Figure 5. COBOL Sample (Part 2 of 4)



IBM COBOL PROGRAM SHEET

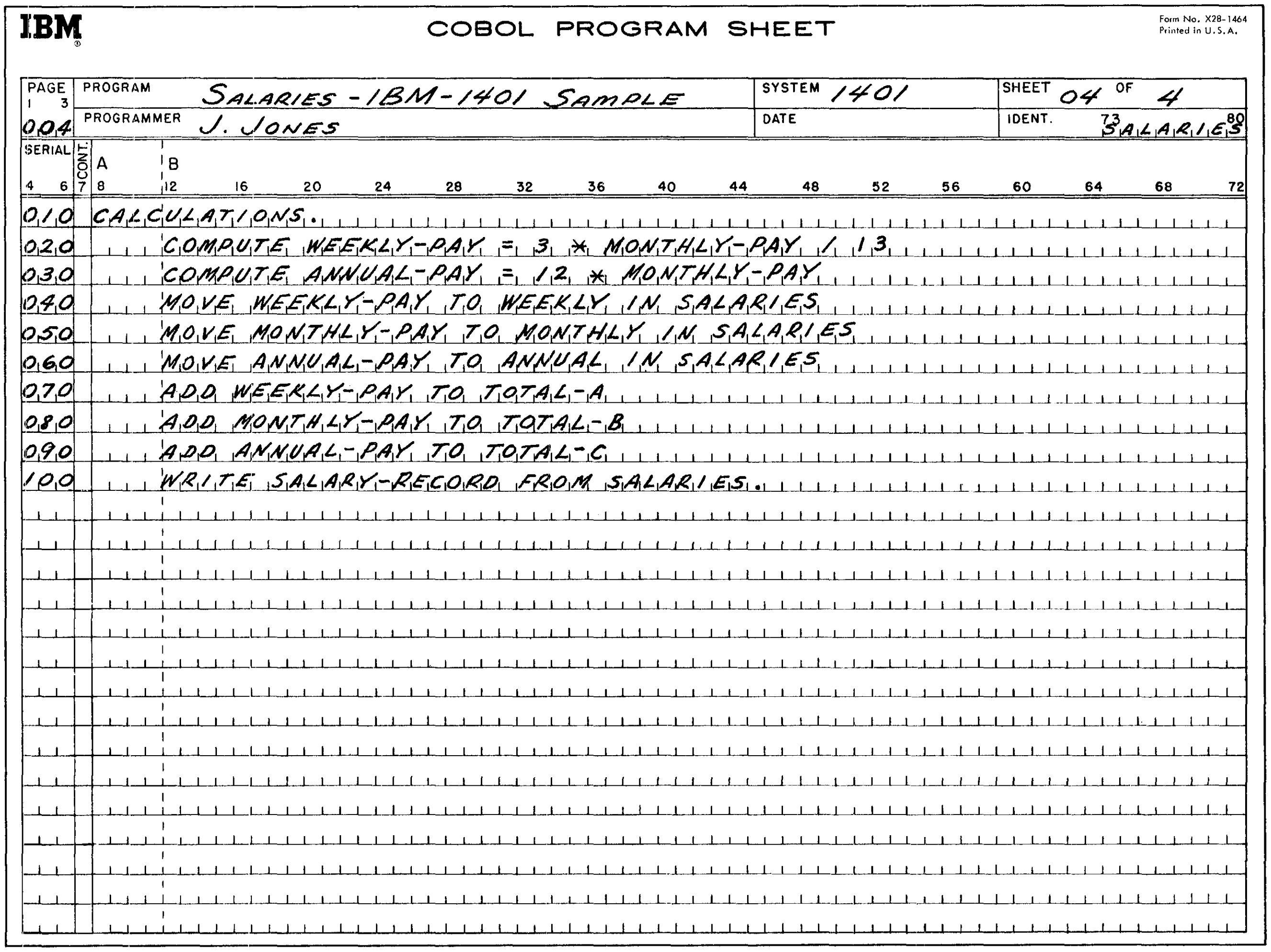
Form No. X28-1464 Printed in U.S.A.

PAGE 003	PROGRAM Salaries - IBM-1401 Sample	SYSTEM 1401	SHEET 03 OF 4
	PROGRAMMER J. Jones	DATE	IDENT. SALARIES

```
SERIAL CONT. A   B
010          1   HEADING.
020          2   FILLER  SIZE 46 ALPHABETIC CHARACTERS   VALUE IS SPACES.
030          2   WEEKLY  SIZE  6 ALPHABETIC        VALUE IS 'WEEKLY'.
040          2   FILLER  SIZE  3 ALPHABETIC              VALUE IS SPACES.
050          2   MONTHLY SIZE  7 ALPHABETIC        VALUE IS 'MONTHLY'.
060          2   FILLER  SIZE  3 ALPHABETIC              VALUE IS SPACES.
070          2   ANNUAL  SIZE  6 ALPHABETIC        VALUE IS 'ANNUAL'.
080          2   FILLER  SIZE 29 ALPHABETIC              VALUE IS SPACES.

090          PROCEDURE DIVISION.
100          START. OPEN OUTPUT SALARY-FILE.
110              WRITE SALARY-RECORD FROM HEADING BEFORE ADVANCING 2 LINES.
120              PERFORM CALCULATIONS
130              VARYING MONTHLY-PAY
140              FROM 500
150              BY   10
160              UNTIL MONTHLY-PAY IS GREATER THAN 1000.
170              IF TOTAL-A = CON-A AND TOTAL-B = CON-B AND TOTAL-C = CON-C
180              MOVE 'TABLE VALUES ARE CORRECT' TO SHOW
190              WRITE SALARY-RECORD FROM MESG AFTER ADVANCING 2 LINES
200              ELSE
210              MOVE 'TABLE VALUES ARE NOT CORRECT' TO PRSNT
220              WRITE SALARY-RECORD FROM DSPY AFTER ADVANCING 2 LINES.
230              CLOSE SALARY-FILE.
240              STOP RUN.
```

Figure 5. COBOL Sample (Part 3 of 4)



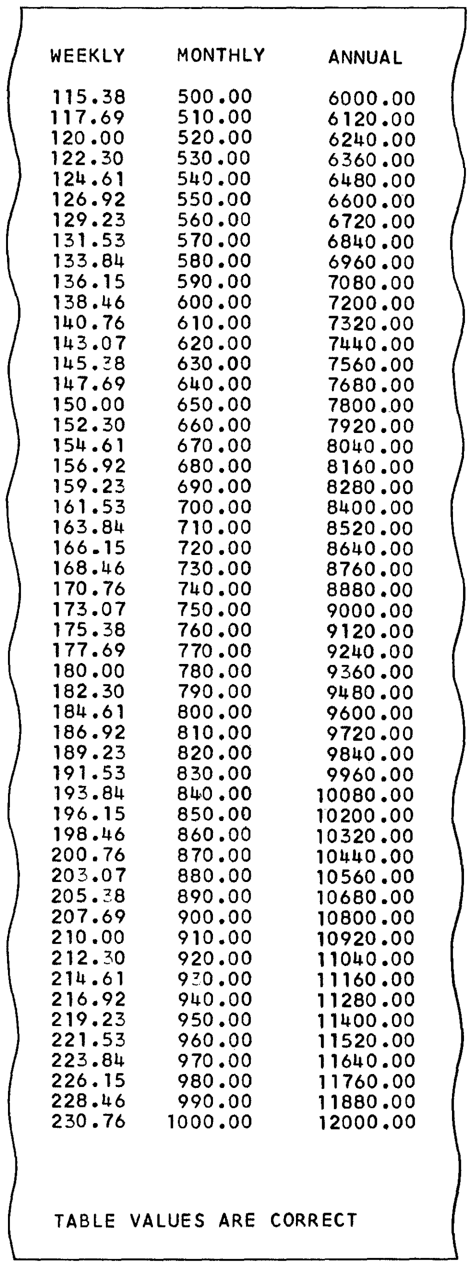
IBM COBOL PROGRAM SHEET

Form No. X28-1464
Printed in U.S.A.

PAGE 1 3: 004	PROGRAM: SALARIES - IBM-1401 SAMPLE	SYSTEM: 1401	SHEET 04 OF 4
	PROGRAMMER: J. JONES	DATE	IDENT. 73 80: SALARIES

```
SERIAL CONT. A   B
4    6  7    8   12
010          CALCULATIONS.
020              COMPUTE WEEKLY-PAY = 3 * MONTHLY-PAY / 13
030              COMPUTE ANNUAL-PAY = 12 * MONTHLY-PAY
040              MOVE WEEKLY-PAY TO WEEKLY IN SALARIES
050              MOVE MONTHLY-PAY TO MONTHLY IN SALARIES
060              MOVE ANNUAL-PAY TO ANNUAL IN SALARIES
070              ADD WEEKLY-PAY TO TOTAL-A
080              ADD MONTHLY-PAY TO TOTAL-B
090              ADD ANNUAL-PAY TO TOTAL-C
100              WRITE SALARY-RECORD FROM SALARIES.
```

Figure 5. COBOL Sample (Part 4 of 4)



```
 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 6. Table of Salaries

IBM®

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

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

COBOL (on Tape) Operating Procedures: IBM 1401 — C24-3146-3

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. Comments and suggestions become the property of IBM.

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

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

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

Other ______________________________

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

COMMENTS:

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

C24-3146-3

fold fold

FIRST CLASS
PERMIT NO. 387
ROCHESTER, MINN.

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

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Systems Development Division
Development Laboratory
Rochester, Minnesota 55901

Attention: Programming Publications, Dept. 425

fold fold

IBM 1401 COBOL Printed in U.S.A. C24-3146-3

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

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]