# GE-225

# INTRODUCTIO TO GECOM

GENERAL ELECTR

COMPUTER DEPARTMENT

# GE-225
# INTRODUCTION TO GECOM

GENERAL ELECTRIC

COMPUTER DEPARTMENT
PHOENIX, ARIZONA

# ERRATA SHEET GE-225 INTRODUCTION TO GECOM

OCT 1962

In the interests of increased efficiency and capability, several improvements have been made to the GECOM system since the publication of the GE-225 Introduction to GECOM manual (CPB 230).

Major changes are mentioned briefly below. More detailed descriptions of these and minor changes are available in the two revised publications:

GE-225 GECOM Language Specifications

GE-225 GECOM Operations Manual

## ADDITIONAL FEATURES

### Compilation

The current configuration of the GECOM system permits program compilation on GE-225 systems having four, five, or six magnetic tape handlers with commensurate reduction in compilation time.

### Relocatable Sections

The GECOM system user can now more readily partition a program into Segments and can thereby compile and test each segment separately. Use of this feature requires an appropriate control routine, which can be a modified version of that used for the main program segment. Segments can be compiled so that they can be relocated in memory when all segments are rejoined into a single program.

### Common-Storage

The COMMON~STORAGE Section of the Data Division has been fully refined to provide for the description of data to be stored in memory locations that are reserved for shared usage by two or more program segments.

### Nested Segments

Provision is made to allow program segments or sections to contain PERFORM sentences which execute other sections.

### "N" Controller Compilation

Compilation can be performed using magnetic tape handlers with one to six magnetic tape controllers, as specified by the GECOM user.

### Sequence Check

At the user's option, source program card sequence numbers can be checked.

### Control Transfers

At the user's option, control transfers based on the type of current record of an input file (determined by automatic Control Key tests) are provided. These transfers are made using statements similar to the following:

1. GO......DEPENDING ON RECORD OF file~name.
2. If record~name GO.....

## SOURCE PROGRAM DECK SEQUENCE

To facilitate many of the above changes and to provide for future improvements and extensions, the organization of the source program deck has been changed slightly. The Data Division must precede the Procedure Division and the END PROGRAM statement (previously at the end of the Data Division) must now be the last statement in the Procedure Division.

Source programs which were previously compiled can be recompiled (if desired) by inserting the Data Division cards, less the END PROGRAM statement, before the Procedure Division and appending a new END PROGRAM statement to the Procedure Division.

## EDITED LIST

Minor changes have been made to the format of the Edited List. For example, the interchanging of Data and Procedure Divisions described above is reflected in the Edited List.

Also, the Edited List now provides a count of 1) the GE-225 words that comprise the required subroutines and supplied program segments, 2) the words generated for the main program, and 3) the total of these two groups of words.

## FUTURE CAPABILITY

Currently under field test is an extension of the GECOM system which enables the compiler to produce object programs utilizing the 16K memory.

# CONTENTS

# APPENDIX 4. GLOSSARY

A list of important terms (most of which are used frequently in the body of this manual and many of which are encountered frequently in other GECOM literature) have been included in this glossary. Most definitions are deliberately brief and are not intended to be comprehensive; many of the terms have additional meanings. For more detailed and more exhaustive listings, the reader is referred to any of several excellent glossaries of information processing terminology.

ADDRESS - A specific location in storage or memory. Actual addresses are numeric. Addresses used in GECOM are symbolic, that is, represented by names.

ARITHMETIC EXPRESSION - A sequence of data names, numeric literals, and/or mathematical functions connected by mathematical symbols.

BCD - Binary Coded Decimal; a system for representing any character of the character set of the computer by a group of binary digits.

BEGINNING FILE LABEL - A group of records (blocks) which identifies a file in a multifile magnetic tape. It is block 0, the first block of each file.

BINARY NUMERIC - A digit or group of characters or symbols representing the total units using the base two: a number expressed in binary digits or bits, 0 and 1.

BLOCK - A group of records read from or written on magnetic tape as a single physical tape record.

BLOCK SIZE - The number of words in a block.

BUFFER - Storage locations used to compensate for differences in rate of data flow when transmitting data from one device to another.

CHARACTER - One of a set of basic symbols used to express data. Includes decimal digits 0 through 9, the letters A through Z, punctuation, and special symbols.

CONDITIONAL EXPRESSION - An expression that can be either true or false.

CONDITIONAL NAME - A name assigned to a possible value of a numeric or alphanumeric field or element. A conditional name must be described in the Data Division.

CONSTANT - A value used in a program without alteration. Constants are either literal, figurative, or numeric in GECOM.

DATA IMAGE - The characteristics of a data field; that is, length, content, sign, and character type for each position. The data image is used within the Data Division to define data input and output.

DATA NAME - A programmer-assigned word naming a file, record, field, constant, or other data. Data names are composed of letters, numerals, and hyphens, not exceeding 12 characters, and may be names of records, groups, fields, arrays, elements, sections, or true-false variables.

ELEMENT - A subdivision of a field. For example, a date field could contain a DAY element, a MONTH element and a YEAR element.

FIELD - A unit of data within a record. It may or may not be a part of a group.

FIGURATIVE CONSTANT - A special name representing specific values [ZERO(S), ZEROES, SPACES, ONE(S), through NINE(S)]. May be used in procedure sentences to imply strings of characters.

FILE - A set of records

FIXED-POINT - A number which includes a decimal point, either between digits or following them (1.23, 123., or 123.0)

FLOATING-POINT - A number expressed as a whole number, a decimal fraction, and a power of ten. (1.287*$10^{-2}$)

GENERATED FIELD - A field (of data) which is generated as a result of calculations and is not input to the program.

INSTRUCTION - A group of symbols causing the data processor to perform some operation.

INTEGER (as used in this manual) - A number of 5 digits or less not containing a decimal point.

# ILLUSTRATIONS

## APPENDIX 3. SOURCE PROGRAM ORDER FOR COMPILATION

| | | |
|---|---|---|
| I. | IDENTIFICATION DIVISION | Mandatory |
| | PROGRAM~ID. | Mandatory |
| | NEXT~PROGRAM | Optional |
| | AUTHOR. | Optional |
| | DATE~COMPILED. | Optional |
| | INSTALLATION. | Optional |
| | SECURITY. | Optional |
| | REMARKS. | Optional |
| II. | ENVIRONMENT DIVISION. | Mandatory (whether or not any sentences follow) |
| | OBJECT~COMPUTER. | Optional |
| | I~O~CONTROL. | Optional |
| | FILE~CONTROL. | Optional |
| | COMPUTATION~MODE. | Optional |
| III. | PROCEDURE DIVISION. | Mandatory |
| | Closed sections and decision tables delimited by BEGIN-END | Placement mandatory if sections are used. |
| | Master program | Mandatory |
| IV. | DATA DIVISION. | Mandatory |
| | ARRAY SECTION. | Optional |
| | TRUE~FALSE SECTION. | Optional |
| | INTEGER SECTION. | Optional |
| | FILE SECTION. | Mandatory* |
| | OUTPUT FILES. | Mandatory* |
| | INPUT FILES. | Mandatory* |
| | WORKING~STORAGE SECTION. | Mandatory* |
| | COMMON~STORAGE SECTION. | Optional |
| | CONSTANT SECTION. | Optional |
| | END PROGRAM. | Mandatory* |

* The section heading card is mandatory; further entries under it are optional.

## SOFTWARE MANUALS

GENERAL ELECTRIC reserves the right to make alterations, advances, or modifications to the existing program for reasons of increased efficiency.

SUMMARY GUIDE FOR DATA DIVISION FORM PREPARATION (continued)

|  |  |  |  |
|---|---|---|---|
|  |  | 11 | 11 |
| Inserts a comma in corresponding field positions. Automatically suppressed by floating dollar signs, zero suppression, asterisk filling. | , |  |  |
| If position occupied by Z in numeric field becomes zero, zero is suppressed and position prints blank. | Z |  |  |
| If position occupied by * becomes zero, * is printed. | * |  |  |
| If position occupied by $ in numeric field becomes zero, move $ into it. | $ $ |  |  |

END PROGRAM. The final entry of the data division must be END PROGRAM starting in column 8 and terminating with a period.

# PREFACE

## ABOUT PROGRAMMING

The programming of information processing systems has traditionally been a costly and time-consuming part of automatic data processing. In the past, many applications that otherwise would readily lend themselves to data processing techniques were avoided because of programming costs. Efforts to improve programming techniques have been directed toward producing faster, more economical, and more accurate programs by placing more of the burden on the data processing equipment.

Various combinations of symbolic coding systems (with one-to-one correlation between machine code and symbolic code), macro-instruction coding systems (with a many-to-one correlation between machine code and macro-code), libraries of standardized subroutines, and other innovations were developed to accelerate programming. Despite these improvements, programmers still prepared programs in terms dictated primarily by the computer; programming languages remained essentially machine-oriented languages.

Today, compiler programs provide the programmer with additional leverage. Program coding can be done in a language more suited to the problem instead of in the purely machine-oriented data processor language.

The GE-225 GECOM system, an advanced and effective automatic coding method, provides the next logical step in programming evolution. GECOM is a step toward fulfillment of the much-needed total systems concept--a concept that deems an information processing system to be an integration of application, programming, and information processor or computer.

The GECOM system is further characterized by its applicability to all classes of information processing problems, its ability to grow, and its inherent provisions for use by future General Electric general-purpose computers. GECOM permits coding in the problem languages of business, science, and industry. GECOM can be adapted to future extensions of existing problem languages as the requirement arises, without obsoleting programs prepared to present specifications.

## ABOUT THIS MANUAL

This manual is presented as a general information manual about the GE-225 GECOM system and is organized to fill the needs of many people having different levels of familiarity with automatic information processing.

For readers with no previous experience in data processing or computer programming, it is suggested that the entire GE manual be covered. Persons having such previous experience, but who are unfamiliar with the GE-225 Information Processing System, are referred to other General Electric publications, listed below.

Readers already familiar with the fundamentals of programming can begin directly with the section, GECOM Programming Language, with no loss in continuity.

Following the section on GECOM programming language is discussion of the Basic GECOM System. All elements are discussed briefly with the intent of providing overall familiarity with all aspects of GECOM.

The next section treats the two major extensions to GECOM, (TABSOL and the Report Writer), which are first mentioned in the GECOM programming language section, but are more effectively discussed after an understanding of GECOM is achieved.

The reader should not assume that reading this manual will make him a master GECOM programmer. The most effective use of GECOM depends upon training and application. More detailed information concerning the various aspects of the GECOM system can be found in the following General Electric publications:

| | |
|---|---|
| GECOM | GE-225 Language Specifications<br>GE-225 General Compiler Operations Manual, CD225H1 |
| TABSOL | GE-225 TABSOL Manual, CPB 147<br>GE-225 Introduction to TABSOL, CPB 147 A |
| GAP | GE-225 Programming Reference Manual, CPB 126 |

## SUMMARY GUIDE FOR DATA DIVISION FORM PREPARATION (continued)

9 11.

| | |
|---|---|
| Position contains an alphabetic character, A-Z, or a blank. | A |
| Position contains an integer 0-9. | 9 |
| Position contains a numeral 0-9 with an 11-row overpunch when negative and no overpunch when positive. | R |
| Position contains a numeral 0-9 with a 12-row overpunch when the field is positive and an 11-row overpunch when the field is negative. | I |
| Indicates an assumed decimal point. Neither the V or the decimal point occupy an actual field position. | V |
| Indicates number following E is a power of ten to which the number preceding the E must be raised. E does not occupy field position. | E |

# ACKNOWLEDGEMENT

"This publication is based in part on the COBOL System developed in 1959 by a committee composed of government users and computer manufacturers. The organizations participating in the original development were:

Air Materiel Command, United States Air Force
Bureau of Standards, Department of Commerce
David Taylor Model Basin, Bureau of Ships, U. S. Navy
Electronic Data Processing Division, Minneapolis-Honeywell Regulator Company
Burroughs Corporation
International Business Machine Corporation
Radio Corporation of America
Sylvania Electric Products, Inc.
Univac Division of Sperry Rand Corporation

In addition to the organizations listed above, the following other organizations participated in the work of the Maintenance Group:

Allstate Insurance Company
Bendix Corporation, Computer Division
Control Data Corporation
DuPont Corporation
General Electric Company
General Motors Corporation
Lockheed Aircraft Corporation
National Cash Register Company
Philco Corporation
Standard Oil Company (N. J.)
United States Steel Corporation

This COBOL-61 manual is the result of contributions made by all of the above-mentioned organizations. No warranty, expressed or implied, is made by any contributor or by the committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or by the committee, in connection therewith.

It is reasonable to assume that a number of improvements and additions will be made to COBOL. Every effort will be made to insure that the improvements and corrections will be made in an orderly fashion, with due recognition of existing users' investments in programming. However, this protection can be positively assured only by individual implementors.

Procedures have been established for the maintenance of COBOL. Inquiries concerning the procedures and methods for proposing changes should be directed to the Executive Committee of the Conference on Data Systems Languages.

. . . . . . . .

Any organization interested in reproducing the COBOL report and initial specifications in whole or in part, using ideas taken from this report or utilizing this report as the basis for an instruction manual or any other purpose is free to do so. However, all such organizations are requested to reproduce this section as part of the introduction to the document. Those using a short passage, as in a book review, are requested to mention "COBOL" in acknowledgment of the source but need not quote the entire section."

SUMMARY GUIDE FOR DATA DIVISION FORM PREPARATION (continued)

7 11

L Literal; no name used. All other columns are completed as for fields and elements.

OTHER OUTPUT RECORD ENTRIES

▨ Not used for output entries.

▨ B or other character forces lower levels with numeric data description (9) to be in standard binary form unless lower level Format indicates non-standard binary data. A blank in column 43 forces BCD data output.

▨ Forces unpacked data to be left
L (L) justified and zero filled or
R right (R) justified and blank filled.

# INTRODUCTION

## WHAT IS GECOM?

The GE-225 GECOM system is an advanced and highly effective method for preparing sets of directions for the GE-225 Information Processing System. As a system, it consists of three elements: Language, Compiler, and Computer. These three terms are further explained below.

### THE LANGUAGE

A language is, in general, a means of communication. In the visual form, it usually consists of a set of symbols (such as our alphabet), which can be arranged into meaningful groups (words). Properly arranged aggregates of these groups or words can communicate ideas, action, commands, and questions.

The direction of an automatic information processing system in the performance of a given operation requires communication between man and machine. Just as communication between two men requires a language intelligible to both, communication between man and machine requires a common language. This common language can be machine-oriented (that is, related closely to the basic means by which the computer accepts and presents information, and requiring tedious translation by man of his directions into machine-acceptable form), or the language can be problem-oriented (enabling man to express directions in a form more convenient to the application and placing the burden of the translation on the computer), or it can lie somewhere between these extremes. Machine-oriented and problem-oriented languages are discussed further in the section, "General Programming Concepts".

The GECOM language is a problem-oriented language designed to handle scientific problems as well as general business information processing. The primary basis for the language structure is COBOL, the COmmon Business-Oriented Language for programming digital computers. COBOL is further discussed in the section, "GECOM Programming Language".

In addition to the capabilities derived from COBOL, GECOM language incorporates many of the features of ALGOL, (an ALGOrithmic Language for stating mathematical computations), such as capabilities to evaluate complex equations, Boolean expressions, and mathematical functions. These computations may be performed in either fixed or floating-point arithmetic.

Further versatility is provided by the incorporation of TABSOL and the Report Writer into the language. TABSOL, for TABular Systems-Oriented Language, is a system for expressing decision logic in a simple tabular form. The Report Writer facilitates report preparation and improves documentation. TABSOL and the Report Writer are discussed in the section, "Extensions to GECOM".

GECOM language is not limited to the language capabilities and the extensions mentioned above. General Compiler versatility permits inclusion of GAP, the basic symbolic language (machine-oriented to a degree) of the GE-225 Information Processing System. GAP, for General Assembly Program, is a straightforward symbolic assembly system for the GE-225.

### THE GENERAL COMPILER

If communication with the computer is to occur in problem-oriented language, some means must be provided to translate that language within the computer into machine-oriented form. A set of directions for a computer, regardless of the language in which it is prepared, is called a program or, sometimes, a routine. A program, manually prepared, is generally termed a source program. A source program which has been translated into a machine-oriented program is an object program. One means of translating a source program into an object program is to use a specially-prepared program (called a compiler) which, within the computer, operates upon the source program as if it were data and transforms it into an object program.

The General Compiler (from which the GECOM system derives its name) is a unique program specifically designed to reduce sharply the traditionally high programming costs associated with the computer applications. GECOM is a highly versatile and dynamic "program generator"; versatile because it accepts source programs written in a variety of languages; dynamic because both the range of languages and the computer types to which it is applicable can

SUMMARY GUIDE FOR DATA DIVISION FORM PREPARATION (continued)

5 11

FL Field literal. Any legal data name. Used for named fields with fixed values. Rules that apply to fields also apply to field literals. Actual value of literal is enclosed in quotation marks in columns 55 through 80.

OUTPUT RECORD ENTRIES:

R Output record-Name in columns 11 through 22; may be qualified by entry of a qualifier in columns 24 through 35. If record name is unique, It need not be qualified.

P Forces all levels within record to be
U packed (P) or unpacked (U) except
binary numerics.

*G *group name in columns 11 through 22. May be qualified. If 2 qualifiers are needed,first goes in columns 24 through 35, second in
~ next line columns 24 through 35 and a tilde in column 7.

P Forces lower levels to be
U packed or unpacked.

## THE INFORMATION PROCESSING SYSTEM

Although the effective use of the GECOM system does ɔt require a detailed knowledge of machine-language programming or data processing systems, some such knowledge is desirable, and perhaps is essential if a valid evaluation of the system is to be made.

Data processing needs have resulted in the development of a great variety of computers. While the physical form and the specific logic flow differ widely, general functions and information flow are similar.

The modern computer or information processor consists of five elements as illustrated in Figure 1: Input, Output, Storage, Arithmetic-Logic, and Control. Communication with the computer is possible only through the input and output elements.

The term, input element, is a functional concept, not the name of a unit of equipment. Only through the input element can data enter the processing system. A system may have one or more of several input media: punched cards, punched paper tape, magnetically-encoded tape, or specially-printed documents. Not all computers have available all input media.

The output element makes it possible for the system to perform a useful function; without an output intelligible to the user, a data processor is useless. Output can take one or more of these forms: punched cards, paper tape, magnetic tape, printing, or any of several special-purpose, machine-controlled forms, such as magnetic-ink encoded (MICR) documents.

Input data must be presented to the system in such a way that the system can manipulate and store it internally. For this reason, data is fed into the system in a form that can be readily converted to the internal electronic language of the system (machine language). Similarly, output data is reconverted to an externally-usable form after processing.

The storage element is functionally subdivided into two general types of storage. One, characterized by limited capacity, high speed, and relatively high cost, is referred to as main storage, memory, core storage, core memory, or simply "core". The latter three terms are popular because tiny magnetic cores are the storage medium in many data processors. The other general type of storage, characterized by high capacity, lower speed, and lower cost, is called auxiliary storage. Auxiliary storage may take almost any form, with punched cards and magnetic tape, discs, and drums being the most common.

The arithmetic-logic element contains the circuits that perform the manipulations of data required by the task or application. It adds, subtracts, multiplies, divides, shifts and rearranges data, and makes decisions, according to the purpose of the program. Capabilities vary widely between different types of computers.

The control element decodes and interprets the stored instructions in proper sequence to achieve the purpose of the program.

In a given compter, it can be difficult to recognize physically the separate storage, control, and arithmetic-logic elements. Functionally, they are separate and distinct elements in all data processing systems and should be so considered. The input and output elements are more readily recognized; more often than not they are packaged as separate units, such as card readers, paper tape readers, document handlers, magnetic tape handlers, card punches, paper tape punches, and printers.

## GENERAL PROGRAMMING CONCEPTS

Programming is essentially the framing of a set of directions for a computer. A set of such directions prepared for, and to be communicated to, a computer to guide and control it for a particular processing task is a program.

A subroutine, on the other hand, is a set of directions that is generally incomplete (by itself) in the sense that it usually is only part of a program. Programs frequently contain subroutines for directing the performance of discrete portions of an overall data processing application.

Programs and subroutines, in turn, consist of instructions, which are basic and are the smallest meaningful part of a program. Thus, instructions are the basic tools of the programmer from which he frames the set of directions a computer is to follow.

The phrase "to direct a computer" indicates communication, and communication implies language. In practice, a programmer may use several languages in preparing programs, depending upon the computer. Digital computers are constructed and organized so that they can accept coded representations of letters and numbers, and interpret them as directions to be followed in processing data. Programming languages generally fall into one of three categories, depending on how closely related they are to the computer requirements for accepting information. These three categories are: machine language, symbolic language, and automatic coding language.

### MACHINE LANGUAGE PROGRAMMING

Perhaps the most important characteristics of modern information processors is the stored-program concept. In the information processor, instructions

GENERAL ELECTRIC

SUMMARY GUIDE FOR DATA DIVISION FORM PREPARATION (continued)

3 11

F Indicates a field of an input record.
Field name is entered in columns 11 through 22.

P Assumes field is packed or unpacked,
U unless it conflicts with a higher level entry (group, record, or file).

1 Assumes one-word binary numeric data. If the data is not integer, a scaling factor must be supplied in the data image columns.

2 Assumes two-word non-standard binary numeric data. If data is not integer, see note above.

S The preceding image is to be used for this entry. Cannot be used if preceding image has a 1 or 2 in column 37.

▨ If any input groups or fields are repeated consecutively, the number of times repeated is entered here.

are held in the storage element along with the data to be processed. This not only permits step-by-step data manipulation -- it enables the machine to manipulate its own instructions as if they were data. Thus, it is possible for a program to modify ~~itself (if prepared with this intention) and selec~~tively repeat desired portions.

All information processing systems have a repertoire of permissible instructions; these vary in number and scope from one machine type to another and between manufacturers. For any given system, however, instructions can be grouped by general function:

1. Arithmetic
2. Decision
3. Input/Output
4. Control

Arithmetic instructions, as the name implies, enable the data processor to perform arithmetic such as addition, subtraction, multiplication, and division.

Decision instructions enable the system to compare certain data with some standard (other data, perhaps, or the status of some data processor element) and select alternate courses of action.

Input and output instructions permit the reading in and writing out of data via peripheral input/output units.

Miscellaneous control instructions vary most widely between machines and depend largely upon machine design. In general, simpler machines require more control instructions to accomplish a given function or process than do more complex machines.

Even in the most complex machine, individual instructions are very simple operations and a number of them must be used in the proper order to perform a given function.

For many reasons, most modern information processors are designed to operate internally in some form of the binary (two-digit) number system, or a binary-based system, rather than the conventional decimal (ten-digit) system. Certain computer elements are bi-stable devices (that is: conducting or nonconducting, on or off, open or closed) with the two possible conditions expressed as "0" and "1", corresponding to "off" and "on", respectively. The "0" and "1" represent the two digits of the binary number system and are commonly called bits, for binary digits. By grouping computer elements and assigning values to them according to their position in the group, all numbers may be expressed in binary numbers; for example:

9 = 1001 18 = 10010 523 = 1000001011

wherein the 1-bits, by virtue of their position, have values corresponding to the powers of two (1, 2, 4, 8, 16, 32, 64, 128, 256, 512, etc. from right to ~~left). The 0-bits, of course, as in the decimal sys~~tem, denote zero value and establish position. Thus, the first 1-bit following the equal sign in the example, 9=1001, has a weight of eight (the third power of two), and the rightmost 1-bit has the weight of one (the zero power of two).

A somewhat similar system permits the representation of alphabetic and special symbols in coded binary form. In fact, the system described so briefly here is only one example of many binary numbering schemes in use and is used primarily to show the concept and illustrate the complexity of programming in a pure machine language. It is rarely necessary to program most modern computers directly in binary or machine language form.

As a final example of machine language programming, a simple routine or program for a hypothetical binary computer is used. Assume that two numbers are in the main storage of the computer at locations arbitrarily called 1000 and 1001. It is desired that the two numbers be added and the result be placed in another storage location, 1002. The binary coding for this program might appear as follows:

```
(1) 00000000001111101000
(2) 00001000001111101001
(3) 00011000001111101010
```

The internal computer circuits would interpret such a program thusly:

(1) Load the contents of storage location 1000 into the arithmetic unit.

(2) Add the contents of storage location 1001 to the contents of the arithmetic unit.

(3) Store the new contents of the arithmetic unit in storage location 1002.

Obviously, pure binary programming is slow and tedious, partly because of the difficulty in keeping track of long strings of bits. One innovation that alleviates this difficulty is the use of an intermediate numbering system between the pure binary and the more familiar decimal system.

If the binary numbers in the example above are grouped into three's, as illustrated below, and repetitively assigned the values of the first three

## SUMMARY GUIDE FOR DATA DIVISION FORM PREPARATION

1 11

DATA DIVISION. Starts in column 8, ends with period. No other entries.

ARRAY SECTION.<br>
TRUE~FALSE SECTION.<br>
INTEGER SECTION.<br>
} Optional sections as required by program. Start in column 8 and end with a period.

FILE SECTION. Identifies characteristics of data in input and output files of the object program. Starts in column 8 and ends with a period. Mandatory section.

OUTPUT FILES. Introduces output file descriptions. Starts in column 8 and ends with period.

INPUT FILES. Introduces input file descriptions. Starts in column 8 and ends with a period.

WORKING~STORAGE SECTION. Introduces working storage descriptions. Starts in column 8 and ends with a period. Mandatory.

COMMON~STORAGE SECTION.<br>
CONSTANT SECTION.<br>
} Optional sections as required by program. Start in column 8 and end with period.

FD File description. Name follows in columns 11 through 22, 12 characters or less.

Descriptions of constants are also accepted by assembly programs. Constants, such as the English word TAX or decimal numbers like 365 are accepted by the assembly program and converted automatically into their machine language equivalents. A legend generally accompanies each description of a constant in the source program to indicate what kind of constant is being described. The legend ALF could be used, for example, to indicate alphabetic constants and DEC for decimal constants.

An assembly program produces the machine language versions of constants and instructions in the object program in such a way that they can be loaded into memory at a later time. Generally, a list is also provided, displaying the symbolic descriptions side-by-side with the output produced in the assembly process for each. The list, called an assembly listing, provides an important documentation of the program. It often contains, also, such aids to program checkout as indications of errors in descriptions and lists of symbolic addresses.

The legends, such as ALF and DEC, that are accepted by the assembly program, but do not stand for actual machine operations, are called pseudo-codes, or pseudo-operations. It is common for an assembly program to provide many of these for the programmer to use. Each extends the ability of the assembly program to prepare or document programs.

The symbolic descriptions of instructions, together with the pseudo-operations that are accepted by an assembly program, constitute what is called an assembly language, or a symbolic language. Although there are numerous exceptions, there is generally one output in machine language for each input in assembly language. For this reason, assembling is often considered to be a one-to-one process.

Symbolic language programming using assembly programs, while considerably simpler and faster than machine language programming, is still highly machine-oriented in that the programmer must have a thorough knowledge of machine-language programming. It is common for source programs written for assembly program processing to result in object programs that are as fast and compact as are equivalent programs prepared directly in machine language. Thus, because symbolic language programs are as efficient as machine language programs, symbolic language programming has almost entirely supplanted the machine language as the basic programming media.

Figure 2 illustrates object program preparation, using an assembly process. First, the programmer prepares the source program in symbolic form, using simple mnemonic codes for the desired machine operations and storage of program constants. Second, the source program is converted to a form suitable for machine entry. The most common representations are hole patterns in punched cards or paper tape or bit patterns on magnetic tape. Usually the programmer prepares his instructions on forms from which a keypunch operator can punch the cards or paper tape for direct entry to the computer or, alternately, for conversion to magnetic tape and the input to the computer.

Next, the assembly program is stored in the computer memory and the source program is input to the computer. The computer, under assembly program control, produces the output -- an object program ready for processing.

At any time after assembiy, the object program, now in machine language form, is input to the computer along with data to be processed. The resultant output -- processed data in the form of punched cards, paper or magnetic tape, or printed reports -- is now ready for use external to the computer.

The assembly system available with the GE-225, as previously mentioned, is known as GAP, for General Assembly Program. For further details, refer to the "GE-225 Programming Reference Manual."

## AUTOMATIC CODING LANGUAGE PROGRAMMING

As pointed out above, the assembly program permits an already-skilled programmer to prepare programs with a minimum of errors by eliminating many of the details of program "housekeeping." It also provides a more readable version of machine language, thus reducing the need for extensive annotation of machine coding. However, it does not eliminate the need for computer and machine language knowledge.

The compiler program permits the programmer to take another large step away from machine-oriented programming and toward problem-oriented language programming. Compiler programs place even more of the burden of object program preparation on the computer by permitting the programmer to state the desired operations in sentence form or in equation form, depending upon the application and the compiler program.

Compilers have several advantages over assembly programs. The language of the compiler is easier for the programmer to learn and easier for him to use, as it is more closely related to his problem. The programmer using a compiler usually does not need as intimate a knowledge of the inner workings of the computer as does the assembly programmer. Programming is faster; the time required to obtain a finished, working program is greatly reduced because there is less chance for the programmer to make a mistake and because most normal errors are detected by the compiler.

# APPENDIX 2. SUMMARY GUIDE FOR GECOM FORM PREPARATION

The following pages briefly summarize the basic rules to be followed in preparing GECOM source programs on the General Compiler Sentence and Data Division Forms. A copy of this appendix is used to provide novice programmers with a convenient guide and a ready reference while becoming familiar with GECOM.

Advanced compilers are not limited to accepting simply symbolic instructions, but can accept statements approximating ordinary English sentences or mathematical equations. Most of these compilers are highly restrictive in the vocabulary and syntax permissible and in the equipment that can be used. The GECOM system is the first to utilize a General Compiler program to permit both English-language and algebraic programming and, at the same time, to embody provisions for structured decision tables and automatic report writing. Additionally, the General Compiler has built-in provision to expand its language capability to encompass other source languages yet to be constructed.

Many of the advantages of compiler programs, particularly those associated with the General Compiler are pointed out in the section, "Advantages of GECOM". Because the balance of this manual is devoted to describing the GECOM system, it would be redundant to further discuss compilers in general.

However, by virtue of the changing requirements placed upon the programmer who may be engaged in GECOM programming, some consideration should be given to his job title.

The average data processing application involves two broad phases. One phase, defining the problem and determining the general method of solution, is generally called systems analysis. The other phase, involving the actual preparation of the program for computer entry, is variously called coding or programming, although in the strict sense coding is only a subordinate part of programming. In some installations, the two phases are performed by separate individuals; in others, both are performed by one person.

The programmer or systems analyst who is thoroughly trained in GECOM principles can communicate more readily with the computer through the General Compiler and, simultaneously, view the overall application in proper perspective. For this reason, the title, systems programmer, is suggested and used in the balance of this manual to describe the GECOM-trained programmer.

WORKING (~STORAGE) - A mandatory Data Division section name.

WRITE - To display a limited amount of information on the console typewriter.

-To release a record or group to an output file.

ZERO(S) - A figurative constant used in procedure sentences.

ZEROES - SAME as ZERO(S)

# GECOM PROGRAMMING LANGUAGE

## GENERAL

All compiler programs accept source programs prepared in specialized language and produce an object program ready for computer processing. Unlike most compilers, GECOM is not restricted to an unduly limited acceptable language. The General Compiler language is actually based on several languages.

The GECOM language evolved primarily from two recent major data processing languages, the business-oriented COBOL and the algorithm-oriented ALGOL. Both languages were developed for solving widely different problems, although from the viewpoint of compiler development they have similar characteristics. These similarities made it possible to provide in one complete and compact package a variety of proven programming techniques. COBOL, which satisfies the needs of the broadest spectrum of data processing applications, provided a basic vocabulary (words and symbols), a basic set of rules of grammer or syntax, and punctuation for clarity. ALGOL, to accommodate the demands of scientific applications, contributes Boolean expressions, floating-point arithmetic, and the ability to express equations concisely.

Many computer applications require neither the extensive file processing facilitated by COBOL, nor the profound mathematics that ALGOL provides, but do involve massive numbers of sequential decisions. To cope effectively with these decisions, General Electric devised structure tables for expressing the relationship of decision parameters. These decision structure tables, and the language in which they are expressed, have been termed TABSOL.

TABSOL has been incorporated into the language accepted by the General Compiler and can be used in combination with the COBOL and ALGOL-like capabilities of GECOM.

In addition to file processing, mathematical applications, and complex decision series, much programming effort is and has been devoted to applications involving report generation. The Report Writer format and language, fully compatible with the General Compiler, gives a fully documented method for preparing reports with minimum programming and debugging effort. The Report Writer is an extension of GECOM and derives much of its advantage from the GECOM system.

Both TABSOL and the Report Writer are discussed in the section, "Extensions to GECOM".

GECOM language is not compartmentalized into the component languages discussed above. In a given source program, it is possible to use COBOL statements containing ALGOL-like algebraic notations; TABSOL decision structure tables can be interspersed with procedure statements; and the Report Writer can be used for report generation. The source program can be prepared using one or all facets of the GECOM language. In addition, if the application so requires, GAP coding sequences can be inserted at will.

## COBOL

Because the GECOM language is based primarily on COBOL, some discussion of COBOL and the history of its development is warranted.

In 1959, a meeting was called in the Pentagon by the Department of Defense to consider the desirability and feasibility of establishing a common language for the adaptation of computers to data processing. Representatives from both users and manufacturers were present. The consensus was that the project was definitely both desirable and feasible. As a result, this Conference on Data Systems Languages (CODASYL) established three committees, Short Range, Intermediate Range, and Long Range, to work in four general areas:

Data Description
Procedural Statements
Application Survey
Usage and Experience

In September, 1959, the Short Range Committee submitted a preliminary framework upon which an effective common business language could be built. After acceptance by the Executive Committee of CODASYL, the report was published in April, 1960, by the Government Printing Office as "COBOL-A

LABEL

LESS

LINE COUNT

LINES

LN - Natural logarithm. A mathematical function that may be used in arithmetic expressions. Calculated in floating-point arithmetic.

LOCK - To prevent a tape from being read or written by program control.

LOG - Common Logarithm. A mathematical function that may be used in arithmetic expressions. Calculated in floating point arithmetic.

LS - LESS than. Used in relational expressions.

MAGNETIC - Part of descriptive name, Magnetic Tape Handler.

MASS - Part of descriptive name, Mass Random Access Data storage.

MEMORY - Main storage, core storage.

MODE - A system of data presentation or processing within the information processing system.

MODULE(S) - Refers to core memory size; one module is 4096 words of storage.

MOVE - To transfer a constant, element, field group, record, or array to a constant, element, etc. of the same size.

MULTIPLE

MULTIPLY - To multiply two quantities and store the result in the last-named field or the specified field.

NEGATIVE

NEQ - Not equal to. Used in relational expressions.

NEXT~PROGRAM - An optional Identification Division sentence name.

NGR - Not Greater Than. Used in relational expressions.

NINE(S) - A figurative constant used in procedure sentences.

NLS - Not Less Than. Used in relational expressions.

NO

NOT - May be used in relational expressions. In logical expressions, it is an exclusive negative.

NOTE - To permit the programmer to write explanatory material in the source program for inclusion in the Edited List, but excluded from the compilation.

OBJECT~COMPUTER - An optional Environment Division sentence name.

OBJECT~PROGRAM - See Glossary

OF

OMITTED

ON

ONE(S) - A figurative constant used in procedure sentences.

OPEN - To initiate the processing of input and output files. Checks or writes labels and does other input-output functions.

OPTIONAL

OR - A logical operator

OUTPUT - A mandatory Data Division section name.

PAGE

PAPER - Pertaining to High-Speed Printer forms.

PERFORM - To cause the specified section to be executed. Control automatically reverts to sentence following the PERFORM.

PLUG(S) - Refers to connectors on the controller selector to which input-output unit controllers are attached.

POSITION

POSITIVE

PRINTER(S) - Pertaining to High-Speed Printer.

PROCEDURE - A GECOM Division name.

PROCEED

PROGRAM - A complete sequence of data processing instructions. May refer to an object program or a source program.

PROGRAM~ID - A mandatory Identification Division sentence name.

# THE BASIC GECOM SYSTEM

## GENERAL

For clarity and simplicity, only the Basic GECOM system is described in this section. Brief descriptions of extensions to Basic GECOM are provided in the section, ''Extension to GECOM''. These extensions, for the most part, expand the capabilities of GECOM to encompass recent language developments.

Implementing a data processing application on a computer involves a broad procedure that has been outlined as follows:

1. Define the problem

2. Determine the procedure to be followed in solving the problem

3. Prepare the computer program, including testing

4. Run the program on the computer with appropriate input data.

If the programmer has at his disposal the automatic coding system of GECOM, the above procedure becomes:

1. Define the problem

2. Determine the procedure to be followed in solving the problem

3. Prepare the source program in problem-oriented language

4. Compile the object program from the source program, using the General Compiler

5. Machine-test (debug) the object program

6. Run the object program on the GE-225 with appropriate input data.

At first glance, automatic coding seemingly complicates the task of data processing. However, as shown in Figure 3, the burden on the programmer is no greater, and often is appreciably less. For example, the step from item 2 to item 3, above, is greatly facilitated by the GECOM-provided ability to express procedural steps in English language statements. Additionally, each statement the programmer writes is several times more powerful than the machine-language or symbolic instructions that he would otherwise use. Also, he is materially assisted in the machine-test or check-out phase, item 5, by the assistance provided by the General Compiler in the form of detailed print-outs of error conditions and of the complete compilation process. The print-outs are as easy to read as the programmer-prepared procedure statements of the source program.

This section is devoted primarily to discussion of item 3, source program preparation, using the GECOM system. Incidental references will be made to the other areas, such as the compilation process, as required.

Assuming that a well-defined data processing problem has been assigned to a systems programmer, he determines the detailed procedures for problem solution and generally prepares a flow chart describing those procedures. Flow charts can be broad or detailed, depending upon the problem and the programmer. Invariably, they are sufficiently detailed to serve as a guide for programming the problem solution. The section, ''Application of Basic GECOM.'' illustrates typical flow charts.

## GECOM SYSTEM COMPONENTS

With these preliminaries out of the way, the programmer is ready to prepare the source program. What does the GECOM system provide him to assist in this task?

First, it provides him the necessary language that eliminates tedious machine-language or symbolic coding. Language is discussed in the following section, ''GECOM Language Elements''.

Second, it provides him with a standard source program organization, which corresponds to the format followed by the compilation output. GECOM source programs are partitioned into four divisions, intended for separate and independent preparation. This facilitates changes; if the procedure must be modified, it can be done with minimal effect upon data parameters; if data changes occur, the data parameters can be changed without affecting the

# APPENDIX 1. THE GENERAL COMPILER VOCABULARY

Words and terms that appear in the following list must be considered to be part of the General Compiler vocabulary and must not be used by the systems programmer in forming data or procedure names, nor may they be used in any manner in a source program other than as provided by the GECOM Language Specifications.

Where warranted, many of the terms have been defined or explained. Terms not so explained were deemed to be self-evident in meaning. In addition, the body of the manual contains many examples that illustrate the use of most of the vocabulary terms.

ABS - Absolute value, or magnitude, of a number, regardless of sign.

ACCESS - Part of descriptive name Mass Random Access Data Storage.

ADD - To add two quantities and store the sum in either the last-named field or the specified field.

ADVANCE - To vertically skip or slew the printer paper.

AFTER

ALL

ALTER - To modify a sequence of operations specified in one or more GO sentences.

AND - A logical operator.

ARE

ARRAY - A multi-valued field that may be referenced by name and subscript. An array may be one, two, or three dimensional and may have corresponding number of subscripts. An array must be defined in the Array Section of the Data Division.

ASSIGN - To direct the placement of a file or program to an input-output media.

ASSIGNMENT - To evaluate an arithmetic expression and assign the result to a field. To equate data names.

ATAN - Are tangent. A mathematical function that may be used within arithmetic expressions. Calculated in floating point arithmetic.

AUTHOR - An optional Identification Division sentence name.

BEGIN - Entrance point to a source program section.

BEGINNING

BGN~FIL~LABL - A tape record preceding each file of a multi-file tape.

BGN~TAP~LABL - The first record on any tape except in multi-file tape.

BINARY - Pertaining to the binary number system, as opposed to decimal or binary coded decimal.

BLOCK - See Glossary

BUFFER - A device which stores data temporarily during transfer operations.

BY

CARD

CLOSE - To terminate processing of input or output reels and files with optional rewind and/or lock.

COMMON (~STORAGE) - An optional Data Division Section name.

COMPUTATION ~ MODE - An optional Environment Division sentence name.

CONSTANT - An optional Data Division section name.

CONTAINS

CONTROL - Interpretation and execution of operations.

CONTROL~KEY - The field or fields by which a record is identified.

COPY - To duplicate from another area.

procedure. In addition, standardization of divisions, sections, procedure statements, and other program elements facilitates communication between programmers and permits program debugging in the same language in which the program was written.

The four divisions of a GECOM source program are:

1. The Identification Division
2. The Environment Division
3. The Data Division
4. The Procedure Division

The Identification Division, Figure 4, provides the programmer with the means for labelling and describing the source program in English-language form. In addition to the program name, author (programmer) and date compiled, this division can include other pertinent information, such as next-program-in-sequence, security classification, location, and explanatory comments as needed. During compilation, this data becomes the label for the object program and is automatically reproduced on output listings, such as the Edited List.

Programmer use of the Identification Division is flexible. The only portion required by the General Compiler is the division name and the PROGRAM ID sentence; all other sentences are at the programmer's option.

Preparation of the Identification Division is discussed further in the section, Application of Basic GECOM.

The Environment Division, Figure 5, provides a link between the source program and the data processing equipment. It defines the computer system configuration and its relationship to the source and object program. The General Compiler depends upon the Environment Division to provide information which associates input and output equipment with the data names for each file to be used in processing. The information in the Environment Division is specified by the systems programmer in English language clauses.

In preparing the Environment Division, the programmer enters the information in a predetermined way. This format is sectionalized under four sentence headings as described below:

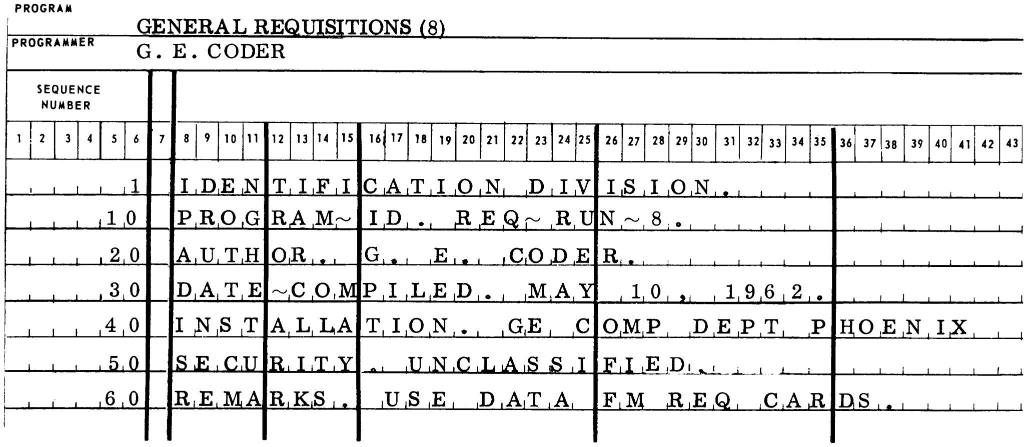
1. The OBJECT~COMPUTER sentence, the first entry, is used to describe the computer on which the object program is to be run.

2. The I~O~CONTROL (input/output control) sentence, the second entry, specifies nonstandard error and tape label checking procedures. In addition, programming control is facilitated by permitting the specification of program rerun points, memory dump assignments, and identification of multifile magnetic tape reels.

3. The third sentence, FILE CONTROL, identifies input/output files and provides for their assignment to specific input/output units.

4. The COMPUTATION~MODE sentence assigns the internal mode of calculation. Sentence use is optional; it is used only when it is desired that computation occur in the floating-point mode, either programmed or in the optional Auxiliary Arithmetic Unit.

The accompanying example illustrates typical entries describing the environment for a representative program. Entry 10 describes the data processing system for which the object program is intended: a GE-225 system with two memory modules (8192 words of core storage), one card reader, one card

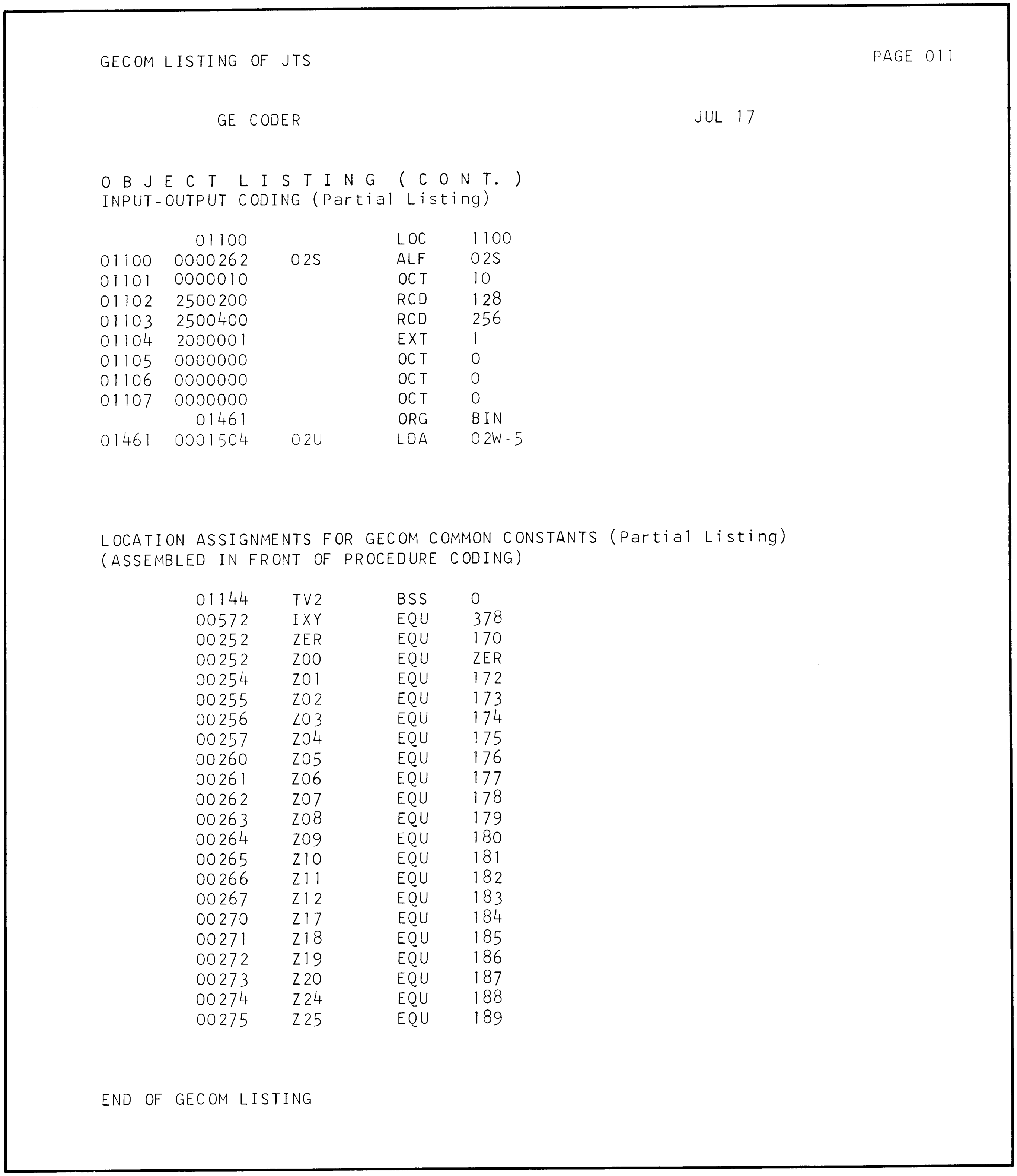


PROGRAM GENERAL REQUISITIONS (8)

PROGRAMMER G. E. CODER

```
SEQUENCE
NUMBER
1 2 3 4 5 6  7  8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
     1        IDENTIFICATION DIVISION.
    10        PROGRAM~ID. REQ~RUN~8.
    20        AUTHOR. G. E. CODER.
    30        DATE~COMPILED. MAY 10, 1962.
    40        INSTALLATION. GE COMP DEPT PHOENIX
    50        SECURITY. UNCLASSIFIED.
    60        REMARKS. USE DATA FM REQ CARDS.
```

Figure 4. Identification Division Layout



```
GECOM LISTING OF JTS                                                  PAGE 011

        GE CODER                                            JUL 17


O B J E C T   L I S T I N G   ( C O N T. )
INPUT-OUTPUT CODING (Partial Listing)

         01100                  LOC    1100
01100   0000262      02S        ALF    02S
01101   0000010                 OCT    10
01102   2500200                 RCD    128
01103   2500400                 RCD    256
01104   2000001                 EXT    1
01105   0000000                 OCT    0
01106   0000000                 OCT    0
01107   0000000                 OCT    0
         01461                  ORG    BIN
01461   0001504      02U        LDA    02W-5



LOCATION ASSIGNMENTS FOR GECOM COMMON CONSTANTS (Partial Listing)
(ASSEMBLED IN FRONT OF PROCEDURE CODING)

         01144       TV2        BSS    0
         00572       IXY        EQU    378
         00252       ZER        EQU    170
         00252       Z00        EQU    ZER
         00254       Z01        EQU    172
         00255       Z02        EQU    173
         00256       Z03        EQU    174
         00257       Z04        EQU    175
         00260       Z05        EQU    176
         00261       Z06        EQU    177
         00262       Z07        EQU    178
         00263       Z08        EQU    179
         00264       Z09        EQU    180
         00265       Z10        EQU    181
         00266       Z11        EQU    182
         00267       Z12        EQU    183
         00270       Z17        EQU    184
         00271       Z18        EQU    185
         00272       Z19        EQU    186
         00273       Z20        EQU    187
         00274       Z24        EQU    188
         00275       Z25        EQU    189


END OF GECOM LISTING
```

Figure 46. Edited List

5. Elements. In a few cases, for convenience, fields are further subdivided into "elements." For example, a part numbering system could be so organized that portions of the part number had added significance. For example: 18253702, NPN Transistor; 18 meaning electrical, 2 meaning a component (not a subassembly), 53 meaning tubes and solid-state devices, and 702 to identify the particular item.

The relationship between these various data levels are readily shown:

```
FILE
    RECORD
        GROUP 1
        GROUP 2
            FIELD
            FIELD
                ELEMENT
                ELEMENT
            FIELD
        GROUP 3
        GROUP 4
```

As mentioned earlier, all data to be used or created by the object program must be defined. A typical Data Division for GECOM is shown in Figure 6, giving representative examples of data definitions. The Data Division for a representative problem is presented and explained in the section, "Application of Basic GECOM". The relationship between Data Division and input data is also shown in Figure 6.

The Procedure Division, Figure 7, indicates the steps that the programmer wishes the object program to accomplish. These steps are expressed in English words, symbols, and sentences that have meaning to the General Compiler. Although the steps described in the Procedure Division closely parallel those of the eventual object program, it is misleading to consider the Procedure Division alone to be the source program. The source program is not complete without Data, Environment, and Identification Divisions.

Sentences in the Procedure Division invariably contain verbs to denote the desired action, names (of data, constants, etc.) or operands to show what is to be acted upon, and various modifiers for clarity. Sentences can be grouped into sections to facilitate reference and permit the performance of a series of sentences out of the normal sequence.

Procedure statements or sentences can be simple:

```
ADD 0.5, RATE OF PAY~FILE.
```

This will create coding in the object program to add the constant 0.5 to whatever value (of the RATE from the PAY~FILE) had been read into the computer. Or statements can be highly complex, involving several clauses and modifiers, such as:

```
IF PART~NUMBER OF MSTR~INVNTRY IS
LESS THAN PART~NUMBER OF TRANSAC-
TIONS GO TO WRITE~MASTER, IF EQUAL GO
TO UPDAT~MASTER, IF GREATER GO
TO NEW~RECORD.
```

This statement would result in object program coding to cause the following:

1. The part number of the master inventory record (previously read in) would be compared with the part number of the current transaction record.

2. If the part number of the master inventory record is:

    a. the lesser of the two, program control is transferred to a routine called WRITE~MASTER, which causes the master inventory record to be written out as part of a master file,

    b. equal to the transaction part number, program control is transferred to a routine called UPDAT~MASTER, which modifies the master inventory record in some manner,

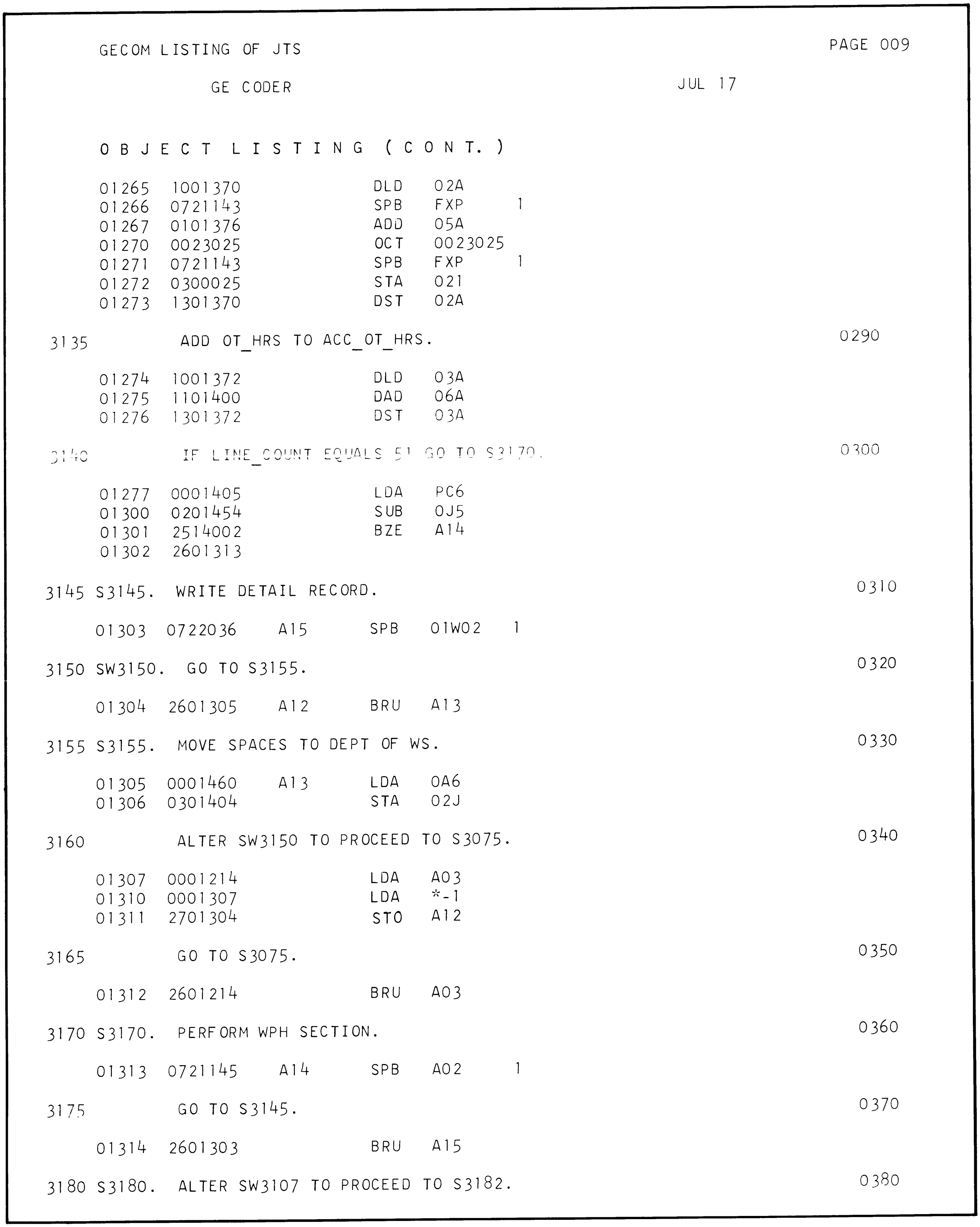
    c. the greater of the two, program control transfers to a routine called NEW~RECORD, which causes a new record to be added to the master file.

Procedure Division sentences are performed in the sequence in which they appear, unless that sequence is modified by a "GO" or a "PERFORM" statement as explained in the next section of this chapter, "GECOM Language Elements".

Typical Procedure Division statements are illustrated in Figure 13. Note that sentences can be named (for reference to them by other sentences) or unnamed. Lines 20, 30 and 70 have been named SENT~1, SENT~2, and SENT~3, although more descriptive names can be assigned at the programmer's discretion. More detailed information for preparing a source program Procedure Division is covered in the section, "Application of Basic GECOM".

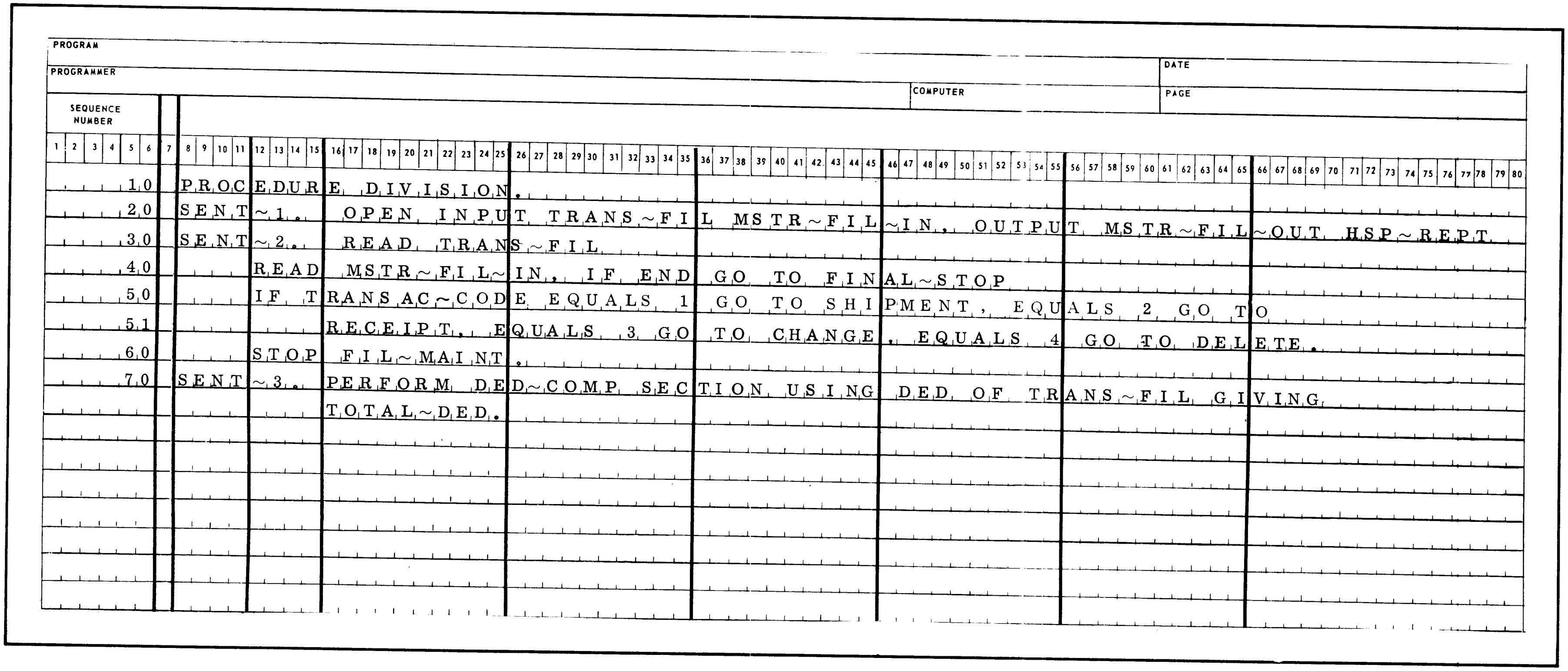
In addition to LANGUAGE and ORGANIZATION, the third item that the GECOM system provides for the programmer is a set of forms to facilitate source program preparation and documentation. Two basic forms are provided, the General Compiler Data Division Form, number CA-14, and the General Compiler Sentence Form, number CA-13.

Both forms are designed to make it easy to translate the programmer-prepared source program information into a machine-readable form, such as punched cards or paper tape. Each horizontal line of either form provides for up to 80 units of information, corresponding to 80 punched card columns.



```
        GECOM LISTING OF JTS                                        PAGE 009

                  GE CODER                                   JUL 17


        O B J E C T   L I S T I N G   ( C O N T . )

        01265  1001370              DLD   02A
        01266  0721143              SPB   FXP       1
        01267  0101376              ADD   05A
        01270  0023025              OCT   0023025
        01271  0721143              SPB   FXP       1
        01272  0300025              STA   021
        01273  1301370              DST   02A

   3135          ADD OT_HRS TO ACC_OT_HRS.                                0290

        01274  1001372              DLD   03A
        01275  1101400              DAD   06A
        01276  1301372              DST   03A

   3140          IF LINE_COUNT EQUALS 51 GO TO S3170.                     0300

        01277  0001405              LDA   PC6
        01300  0201454              SUB   0J5
        01301  2514002              BZE   A14
        01302  2601313

   3145 S3145.   WRITE DETAIL RECORD.                                     0310

        01303  0722036    A15       SPB   01W02     1

   3150 SW3150.  GO TO S3155.                                             0320

        01304  2601305    A12       BRU   A13

   3155 S3155.   MOVE SPACES TO DEPT OF WS.                               0330

        01305  0001460    A13       LDA   0A6
        01306  0301404              STA   02J

   3160          ALTER SW3150 TO PROCEED TO S3075.                        0340

        01307  0001214              LDA   A03
        01310  0001307              LDA   *-1
        01311  2701304              STO   A12

   3165          GO TO S3075.                                             0350

        01312  2601214              BRU   A03

   3170 S3170.   PERFORM WPH SECTION.                                     0360

        01313  0721145    A14       SPB   A02       1

   3175          GO TO S3145.                                             0370

        01314  2601303              BRU   A15

   3180 S3180.   ALTER SW3107 TO PROCEED TO S3182.                        0380
```

**Figure 44. Edited List**



PROGRAM

PROGRAMMER

DATE

COMPUTER

PAGE

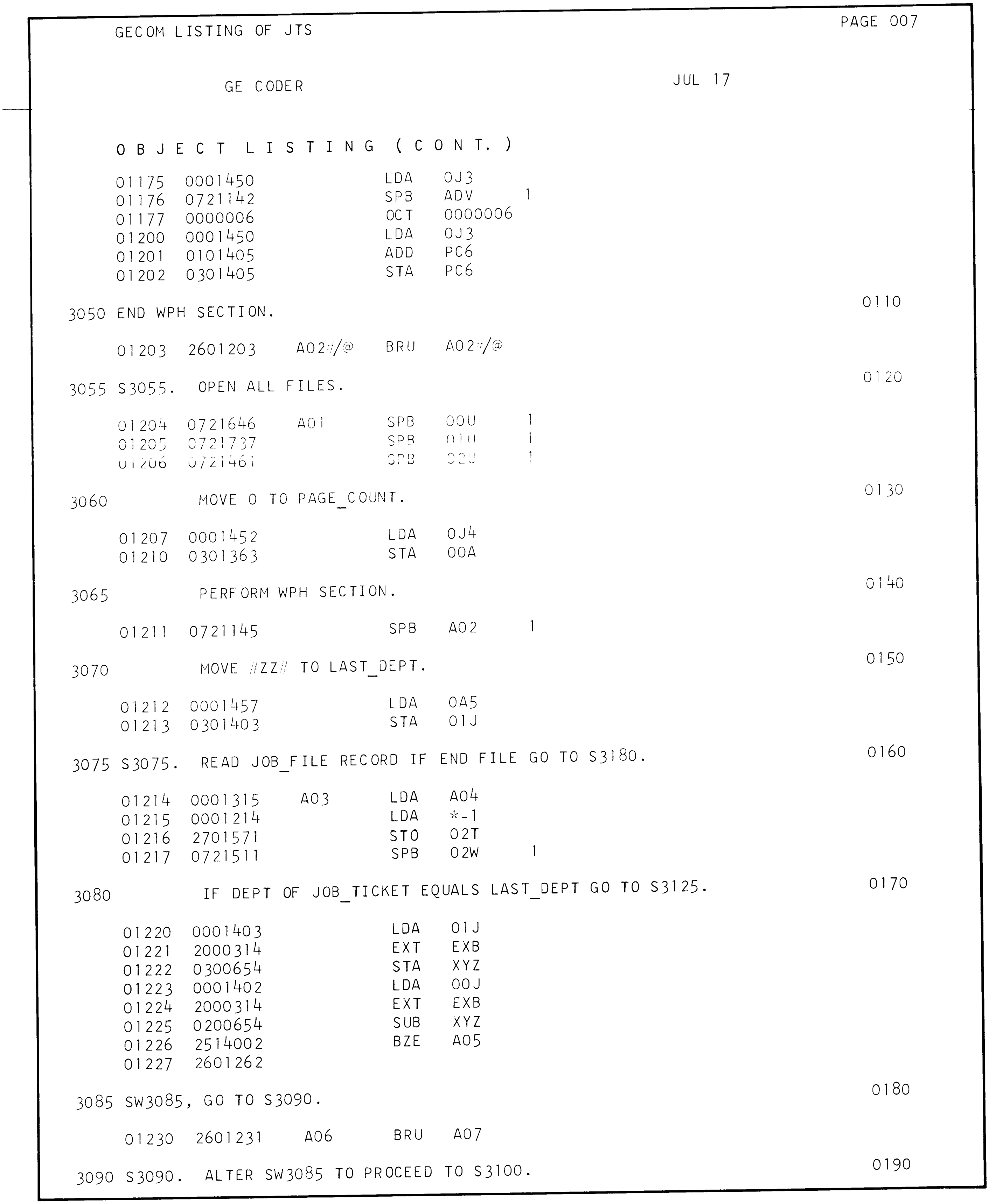
SEQUENCE NUMBER

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

```
    10 PROCEDURE DIVISION.
    20 SENT~1.  OPEN INPUT TRANS~FIL MSTR~FIL~IN, OUTPUT MSTR~FIL~OUT HSP~REPT.
    30 SENT~2.  READ TRANS~FIL
    40     READ MSTR~FIL~IN, IF END GO TO FINAL~STOP
    50     IF TRANSAC~CODE EQUALS 1 GO TO SHIPMENT, EQUALS 2 GO TO
    51         RECEIPT, EQUALS 3 GO TO CHANGE, EQUALS 4 GO TO DELETE.
    60     STOP FIL~MAINT.
    70 SENT~3. PERFORM DED~COMP SECTION USING DED OF TRANS~FIL GIVING
               TOTAL~DED.
```

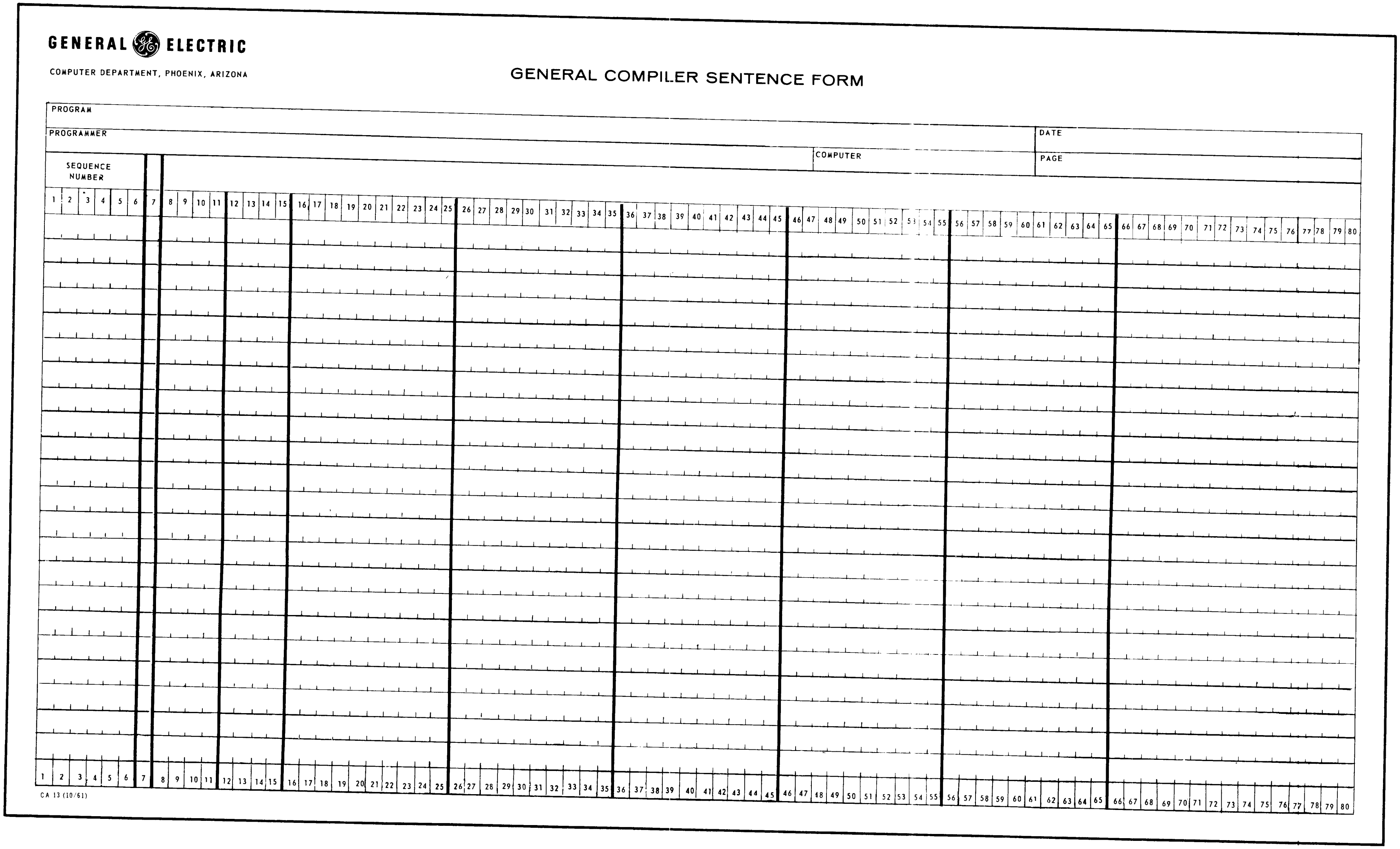
Figure 7. Procedure Division Layout

]



```
        GECOM LISTING OF JTS                                             PAGE 007

                  GE CODER                                    JUL 17

        O B J E C T   L I S T I N G   ( C O N T. )

        01175  0001450               LDA   0J3
        01176  0721142               SPB   ADV        1
        01177  0000006               OCT   0000006
        01200  0001450               LDA   0J3
        01201  0101405               ADD   PC6
        01202  0301405               STA   PC6

   3050 END WPH SECTION.                                                 0110

        01203  2601203    A02#/@     BRU   A02#/@

   3055 S3055.  OPEN ALL FILES.                                          0120

        01204  0721646    A01        SPB   00U        1
        01205  0721737               SPB   01U        1
        01206  0721461               SPB   02U        1

   3060             MOVE 0 TO PAGE_COUNT.                                0130

        01207  0001452               LDA   0J4
        01210  0301363               STA   00A

   3065             PERFORM WPH SECTION.                                 0140

        01211  0721145               SPB   A02        1

   3070             MOVE #ZZ# TO LAST_DEPT.                              0150

        01212  0001457               LDA   0A5
        01213  0301403               STA   01J

   3075 S3075.  READ JOB_FILE RECORD IF END FILE GO TO S3180.            0160

        01214  0001315    A03        LDA   A04
        01215  0001214               LDA   *-1
        01216  2701571               STO   02T
        01217  0721511               SPB   02W        1

   3080             IF DEPT OF JOB_TICKET EQUALS LAST_DEPT GO TO S3125.  0170

        01220  0001403               LDA   01J
        01221  2000314               EXT   EXB
        01222  0300654               STA   XYZ
        01223  0001402               LDA   00J
        01224  2000314               EXT   EXB
        01225  0200654               SUB   XYZ
        01226  2514002               BZE   A05
        01227  2601262

   3085 SW3085, GO TO S3090.                                             0180

        01230  2601231    A06        BRU   A07

   3090 S3090.  ALTER SW3085 TO PROCEED TO S3100.                        0190
```

Figure 42. Edited List



GENERAL ELECTRIC

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL COMPILER SENTENCE FORM

PROGRAM

PROGRAMMER

DATE

COMPUTER

PAGE

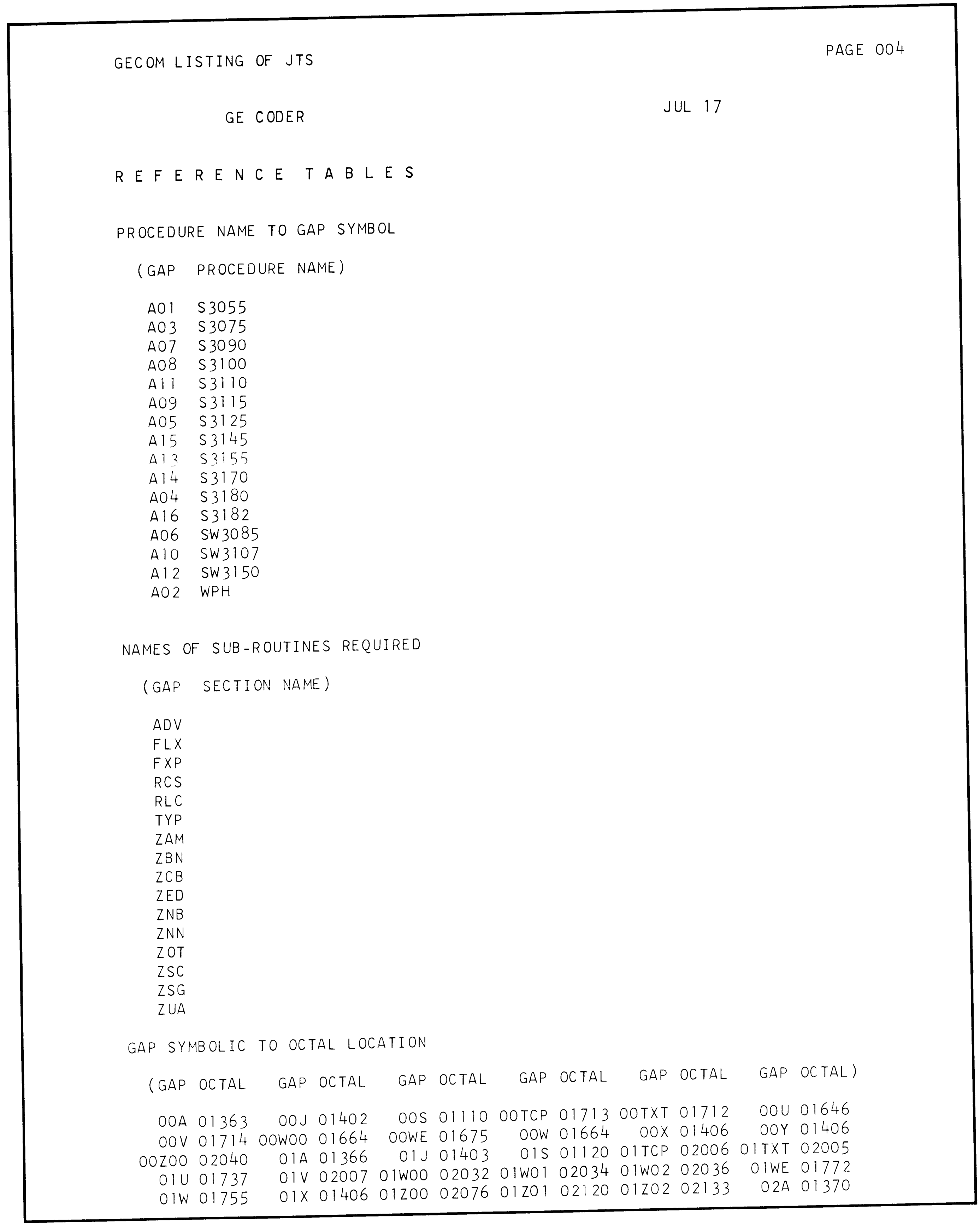
SEQUENCE NUMBER

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

CA 13 (10/61)

Figure 9. The GECOM Sentence Form



```
GECOM LISTING OF JTS                                          PAGE 004

          GE CODER                                   JUL 17

REFERENCE TABLES

PROCEDURE NAME TO GAP SYMBOL

  (GAP  PROCEDURE NAME)

   A01  S3055
   A03  S3075
   A07  S3090
   A08  S3100
   A11  S3110
   A09  S3115
   A05  S3125
   A15  S3145
   A13  S3155
   A14  S3170
   A04  S3180
   A16  S3182
   A06  SW3085
   A10  SW3107
   A12  SW3150
   A02  WPH

NAMES OF SUB-ROUTINES REQUIRED

  (GAP  SECTION NAME)

   ADV
   FLX
   FXP
   RCS
   RLC
   TYP
   ZAM
   ZBN
   ZCB
   ZED
   ZNB
   ZNN
   ZOT
   ZSC
   ZSG
   ZUA

GAP SYMBOLIC TO OCTAL LOCATION

  (GAP OCTAL    GAP OCTAL    GAP OCTAL    GAP OCTAL    GAP OCTAL    GAP OCTAL)

    00A 01363     00J 01402     00S 01110 00TCP 01713 00TXT 01712     00U 01646
    00V 01714 00W00 01664    00WE 01675     00W 01664     00X 01406     00Y 01406
  00Z00 02040     01A 01366     01J 01403     01S 01120 01TCP 02006 01TXT 02005
    01U 01737     01V 02007 01W00 02032 01W01 02034 01W02 02036    01WE 01772
    01W 01755     01X 01406 01Z00 02076 01Z01 02120 01Z02 02133     02A 01370
```

Figure 40. Edited List

The Data Division Form, Figure 8, is used exclusively for describing data to be used in the object program. Headings are provided to guide the proper placement of data. These are discussed in the later section, Data Division Preparation.

The Sentence Form, Figure 9, is used for the preparation of data for the Identification, Environment, and Procedure Divisions. Headings, which would add little, are omitted. Rules for Sentence Form preparation are few and simple.

Where applicable, such rules are discussed in the section, "Application of Basic GECOM," along with the preparation of the four divisions of the source program. The fourth major tool provided by the GECOM system, is the General Compiler itself. Examination shows considerable similarity between the General Compiler program and a complex business data processing object program.

1. The General Compiler operates upon input: the source-language program.

2. Compiler processing consists of repetitive runs of a set of instructions: the General Compiler.

3. It produces an output: the object program.

4. It produces reports: the Edited List and error messages.

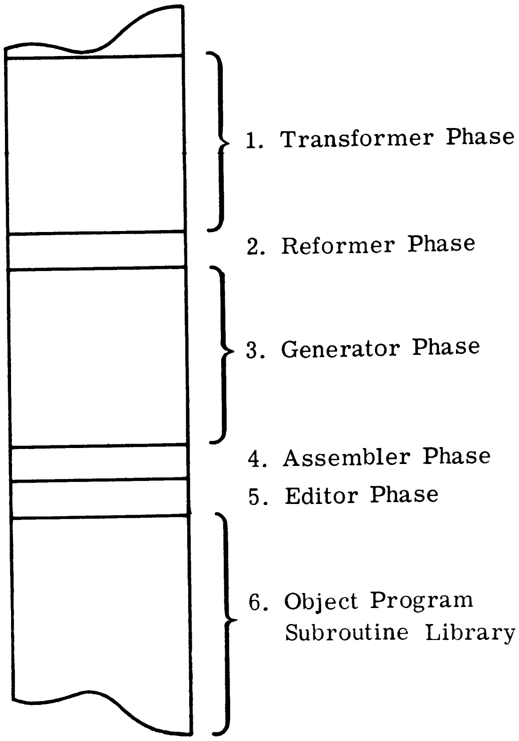


**Figure 11. General Compiler Program Organization**

Figure 10 illustrates, in broad terms, the relationships between the programmer-produced source programs, the General Compiler, the computer, and the output object program.

Up to this point, the General Compiler has been discussed as if it were a single program, and it can still be considered as such. Conversely, it can also be considered to be a series of sequential programs as illustrated in Figure 11. Note that there are five major groupings: Transformer, Reformer, Assembler, Editor, and Subroutines.

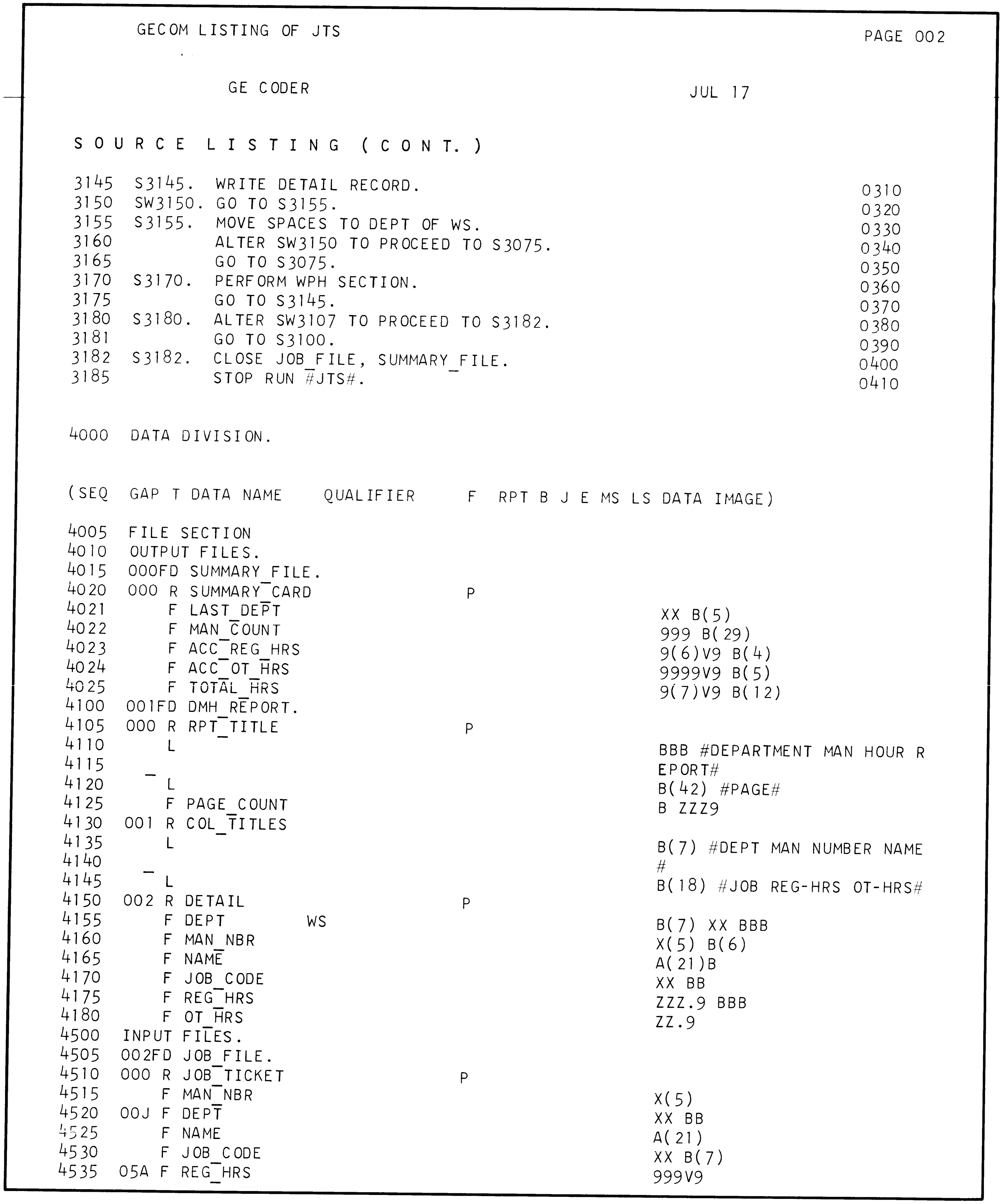
The transformer phase translates the source program into an intermediate internal language suitable for processing, prints out Identification and Environment Divisions as required, groups and organizes Procedure and Data Division material for further processing while checking for validity and consistency, prints error messages, screens out unessential optional words, and initiates the preparation of the object program.

The reformer phase is essentially executive in that it calls forth from the generator library (also a part of the Compiler) those routines required to produce the object program.

The assembler phase translates from the intermediate language, assembles the coding into machine language, and produces the completed object program either in punched cards or on magnetic tape.

The editor phase provides the documentation of the program in the form of the Edited List. This includes a print-out of the entire original source program, a merged list showing the generated symbolic coding and the machine-language coding, and cross-reference tables. Additionally, it lists, from the master list of subroutines below, those required to complete the object program. Examples of the Edited List are included in the section, "Application of Basic GECOM."

The subroutine library is a collection of previously-prepared subroutines common to most object programs that may be required to complete the object program. While these could be produced during compilations, to reduce compilation time and avoid repetitive processing during compiling, the General Compiler shows (on the Edited List) all such subroutines which will be needed when the object program is run. A special program loading routine will place into memory the object program and the



```
        GECOM LISTING OF JTS                                                PAGE 002

                    GE CODER                                        JUL 17

    S O U R C E   L I S T I N G   ( C O N T. )

    3145  S3145.  WRITE DETAIL RECORD.                                    0310
    3150  SW3150. GO TO S3155.                                            0320
    3155  S3155.  MOVE SPACES TO DEPT OF WS.                              0330
    3160          ALTER SW3150 TO PROCEED TO S3075.                       0340
    3165          GO TO S3075.                                            0350
    3170  S3170.  PERFORM WPH SECTION.                                    0360
    3175          GO TO S3145.                                            0370
    3180  S3180.  ALTER SW3107 TO PROCEED TO S3182.                       0380
    3181          GO TO S3100.                                            0390
    3182  S3182.  CLOSE JOB_FILE, SUMMARY_FILE.                           0400
    3185          STOP RUN #JTS#.                                         0410


    4000  DATA DIVISION.


    (SEQ  GAP T DATA NAME     QUALIFIER     F  RPT B J E MS LS DATA IMAGE)

    4005  FILE SECTION
    4010  OUTPUT FILES.
    4015  000FD SUMMARY_FILE.
    4020  000 R SUMMARY_CARD                P
    4021      F LAST_DEPT                                     XX B(5)
    4022      F MAN_COUNT                                     999 B(29)
    4023      F ACC_REG_HRS                                   9(6)V9 B(4)
    4024      F ACC_OT_HRS                                    9999V9 B(5)
    4025      F TOTAL_HRS                                     9(7)V9 B(12)
    4100  001FD DMH_REPORT.
    4105  000 R RPT_TITLE                   P
    4110      L                                               BBB #DEPARTMENT MAN HOUR R
    4115    _                                                 EPORT#
    4120      L                                               B(42) #PAGE#
    4125      F PAGE_COUNT                                    B ZZZ9
    4130  001 R COL_TITLES
    4135      L                                               B(7) #DEPT MAN NUMBER NAME
    4140    _                                                 #
    4145      L                                               B(18) #JOB REG-HRS OT-HRS#
    4150  002 R DETAIL                      P
    4155      F DEPT          WS                              B(7) XX BBB
    4160      F MAN_NBR                                       X(5) B(6)
    4165      F NAME                                          A(21)B
    4170      F JOB_CODE                                      XX BB
    4175      F REG_HRS                                       ZZZ.9 BBB
    4180      F OT_HRS                                        ZZ.9
    4500  INPUT FILES.
    4505  002FD JOB_FILE.
    4510  000 R JOB_TICKET                  P
    4515      F MAN_NBR                                       X(5)
    4520  00J F DEPT                                          XX BB
    4525      F NAME                                          A(21)
    4530      F JOB_CODE                                      XX B(7)
    4535  05A F REG_HRS                                       999V9
```

Figure 38. Edited List

required subroutines which the operator has previously extracted from the library of subroutines provided. At the user's option, required subroutines can be appended to the object program automatically or manually during compilations.

## GECOM LANGUAGE ELEMENTS

Because the GECOM system was developed with COBOL in mind as the basic programming language, the GECOM language elements most closely resemble those of the COBOL language. Also, because the intent is to provide English-language programming, GECOM elements parallel those of English.

GECOM has a basic vocabulary consisting of words and symbols; it has rules of grammar or syntax; and it has punctuation symbols for clarity. In each case, there is greater simplicity than in English: the vocabulary is small; the rules of grammar are simple, yet precise; the use of punctuation is limited. These are true because the demands placed upon the user are kept simple and unambiguous. The source programming language is required to state facts and give instructions clearly and specifically; it is a language of command, not narration, and thus consists primarily of verbs and nouns. These can be formed into simple and complex sentences usually intelligible without special training, although sentences acceptable to the General Compiler cannot be written without familiarity with the grammar.

Words and symbols are the tools of the GECOM programmer and are composed of individual letters, numbers, and special characters. The basic character set of GECOM and equivalent GE-225 character codes are illustrated in the accompanying table, Figure 13. Special character sets are available for the printer.

Many of the basic characters, in addition to being used in words, have special meanings for GECOM; these will be discussed where appropriate.

Words, in GECOM, are divided into two major groups - names and verbs.

### VERBS

As in English, verbs denote action; unlike English, GECOM verbs are never taken in the passive voice, the narrative or declarative sense, or in any tense other than the present tense. Each verb that the programmer uses in the source program (except the verb NOTE) will have some effect in the object program.

Most verbs will be reflected directly in the machine-language coding of the compiled object program; others do not appear in the object program, but do act with the compiler to construct the object program.

Certain words that, in English, are not verbs are considered as such by the General Compiler. The most commonly-used and most useful of these is the word, IF, which is used in expressing conditions, relationships, and comparisons. For example, in the expressions:

```
IF NOT END OF FILE, GO TO . . . . . . . . . .
              OR
IF A EQUALS B, GO TO . . . . . . . . . . . .
```

IF causes a comparison between the actual condition and the stated END OF FILE condition or, in the second example, causes a comparison between A and B. Such near-verbs will be discussed as if they were verbs.

The GECOM verbs and examples of how each might be used are listed in Figure 14.

### NAMES

Most words in the GECOM source program will be names. The programmer is preparing a program for handling data, but is not concerned with the actual data itself; he is more concerned with preparing data manipulation procedures, but once they are written they are only of as much importance as the data they manipulate. For these reasons, and to take advantage of the leverage that GECOM provides, the programmer will refer to data and previously written procedures by name whenever possible.

Names can be readily grouped by type and fall within these groups:

1. Data Names
2. Procedure Names
3. Conditional Names
4. Constants

### DATA NAMES

Data names represent data to be used in an object program, and are programmer-assigned, not to specific data, but to kinds of data. For example, in a file processing application, data names would be assigned to all input and output files, such as:

MASTER~FILE<br>
TRANSACTIONS
PRINT~FILE
etc.

and, within a file, records would bear data names, such as:

STOCK~RCD
PAY~RCD
INV~RCD~1
etc.

The Object Listing includes an "Input/Output Coding" print-out showing all input/output file tables, control coding, and service routines. A complete listing of this subsection for the sample problem requires 439 line entries. Part of the Input/Output Coding list is shown in Figure 46.

The final print-out of the Object Listing and the Edited List is "Location Assignments for GECOM Common Constants," Figure 46. This print-out contains the memory locations for object program constants and the compiler-assigned symbols for the constants. For the sample problem, the complete constant listing contains 138 entries.

| VERB | EXAMPLE |
|---|---|
| ADD | ADD TOTL~RECVD TO ON~HAND~QTY |
| ADVANCE | ADVANCE PAY~REGISTER 20 LINES<br>(to slew or skip printer paper) |
| ALTER | ALTER SENT~25 TO PROCEED TO SENT~33.<br>(to change a previously established sequence of operations.) |
| =(Assignment) | QTY~ON~HAND = OLD~QTY + NO~RECVD<br>(to assign an evaluated arithmetic expression to a specified field) |
| CLOSE | CLOSE PAYROL~FILE<br>(to terminate processing of a file) |
| DIVIDE | DIVIDE NUMBER INTO TOTAL GIVING AVERAGE |
| ENTER | ENTER GAP AT ROUTINE~3<br>(to permit insertion of General Assembly Program coding in a GECOM source program.) |
| EXCHANGE | EXCHANGE OLD~TAX, NEW~TAX<br>(to transpose the contents of two fields) |
| GO | GO TO SENT~10<br>(to depart from the normal sequence of operations) |
| IF | IF LINE~COUNT EQ 58 GO TO ADVANCE~PAGE.<br>(to test a condition and transfer to another operation if condition is satisfied) |
| MOVE | MOVE TOTAL TO SAVE~AREA<br>(to transfer data to another location) |
| MULTIPLY | MULTIPLY 0.18 BY PAY GIVING TAX |
| NOTE | NOTE THIS SENTENCE IS USED FOR CLARITY.<br>(to permit insertion of explanatory text not intended for compilation) |
| OPEN | OPEN ALL INPUT FILES<br>(to initiate file processing) |
| PERFORM | PERFORM FICA~COMP SECTION<br>(to cause execution of a routine in the desired sequence and then return to the sentence following the PERFORM statement.) |
| READ | READ TIME~CARD RECORD<br>(to make input file records available to the program) |
| STOP | STOP<br>(to halt processing of the object program permanently or temporarily.) |
| SUBTRACT | SUBTRACT RECEIPTS OF TRANSAC~FILE FROM ON~ORDER~QTY OF ORDER~FILE GIVING ADJ~ORDER~QTY, IF SIZE ERROR GO TO ZERO~RTN. |
| VARY | VARY CHK~AMT FROM 1 BY 1 UNTIL CHK~AMT GR 5<br>(to initiate and control the repeated execution of the sentence it precedes.) |
| WRITE | WRITE RECORD~1 OF FILE~6<br>(to permit output of data) |

**Figure 14. GECOM Verbs**

3185 STOP RUN "JTS"

This statement is used to generate object program coding for halting processing. In the form used here, the results will be

1. Program halts
2. END is printed by the console typewriter.
3. The literal "JTS" is printed by the console typewriter.

IDENTIFICATION DIVISION PREPARATION

This division enables the programmer to label the source program and provide program identification in the output Edited List.

The Identification Division is prepared on the General Compiler Sentence Form, as illustrated in Figure 35.

Entries for the Job Ticket Summary problem are explained:

1000 IDENTIFICATION DIVISION.

This mandatory heading indicates that entries following are for program identification only. The name should begin in column 8 and be followed by a period.

1005 PROGRAM~ID. JTS.

This entry is mandatory; the name, PROGRAM~ID, should appear beginning in column 8 and followed by a period. The actual program name, JTS, can consist of up to nine typewriter characters followed by a blank, a comma, or a period and can be indented any number of spaces. This name will appear as part of the heading of each page of the Edited List.

1010 AUTHOR. GE CODER

This entry is optional. If used, the sentence name should start in column 8 and be followed by a period. The sentence can be indented as desired, contain up to 30 BCD characters, and ended with a period. If provided, the author's name appears on each page of the Edited List.

1015 DATE COMPILED. JUL. 17

This entry is optional. It can contain up to 30 characters followed by a period. If provided, the compilation date appears on each page of the Edited List.

1020 INSTALLATION. . . .
1025 REMARKS. . . . .

These two sentences, as well as a NEXT~PROGRAM and a SECURITY sentence, are optional. If used, they can contain any information that the programmer wants to appear in the Edited List.

The Identification Division has no effect upon the compilation of the object program, other than that of appearing in the Edited List as described.

## PRODUCING THE OBJECT PROGRAM

Upon completion of the GECOM forms for the source program, the data forms are transcribed to standard punched cards to form the source program deck and organized as shown in Figure 36.

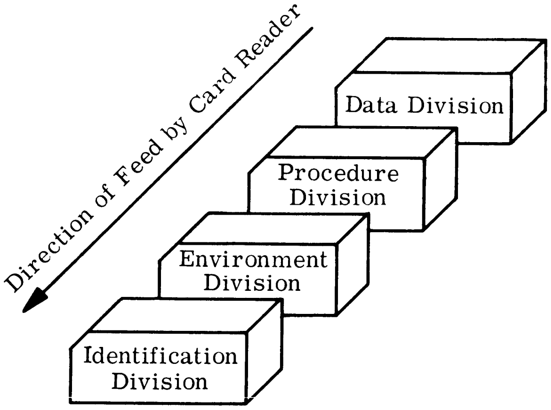


**Figure 36. Source Program Deck Organization**

A special GECOM call deck is placed before the source program deck and the cards are ready for input to the GE-225 via the card reader.

The minimum GE-225 system configuration for compiling the source program is:

GE-225 Central Processor (with 8192 words of core storage)
Console Typewriter
Card Reader
Card Punch
High-Speed Printer
Magnetic Tape Controller
Four Magnetic Tape Handlers
Five Magnetic Tape Handlers (optional)
Six Magnetic Tape Handlers (optional)

The GECOM Master Tape is mounted on the first magnetic tape handler on the system and includes a library of subroutines that might be required to complete the compiled object program. The source

In preparing the source program, the programmer may have difficulty in keeping track of codes that of themselves have no meaning. To provide a reference term, he can assign names to them, thusly:

HOURLY = 0<br>
WEEKLY = 1<br>
MONTHLY = 2

Once names are assigned, they can be used in procedure statements within the source program. Such names as those described above are called conditional names for convenience. In actuality, they are special data names, and are formed subject to the same limitations.

CONSTANTS

Data names are generally assigned by the systems programmer to kinds of data, rather than to specific values, because the actual value of the data named is generally a variable (from record to record, for example) or possibly an unknown to be computed by the object program.

Occasionally (even frequently), the programmer will need to place various kinds of specific data in the program - data which remain the same throughout the program. Such constants are designated as literal constants, numeric constants, and figurative constants.

Literal constants are those the programmer intends to use in the program exactly as written. They may be any combination of up to 30 (or 83, depending upon where used) letters, numbers, and symbols of the GECOM character set. To distinguish them from other names, they must be enclosed in quotation marks:

MOVE “FILE∼NAME” TO COLUMN∼HD.

Literals can be used in output fields to generate headings. They cannot be used in arithmetic calculations.

Numeric constants are comprised of the numerals 0 through 9, plus or minus sign, the letter E for floating-point, and a decimal point. They can be used in three forms of arithmetic calculations: fixed-point, integer, and floating-point.

Fixed-point numerics can contain up to 11 digits, excluding plus or minus sign, and a decimal. Typical fixed-point numerics are:

+2.308 -853.001<br>
0.03 9.11

Integers must not exceed 5 digits:

2308 85300<br>
3 911

For floating-point computations, numerics can be written with mantissas of up to nine digits (one of which must be the left of the decimal) and an exponent between +75 and -75. The largest and smallest floating-point numbers that can be represented are, respectively:

9.99999999E+75 and 0.00000001E-75

If any numeric constant is enclosed in quotation marks, it loses its numeric value and becomes a literal constant.

The constants, 0 through 9 and space (or blank) have been defined within the General Compiler and assigned names. This permits the programmer to use the names within his source program without defining them. These pre-named constants are called figurative constants and are:

0 ZERO or ZEROES<br>
SPACES<br>
1 ONE(S)<br>
2 TWO(S)<br>
3 THREE(S)<br>
4 FOUR(S)<br>
5 FIVE(S)<br>
6 SIX(ES)<br>
7 SEVEN(S)<br>
8 EIGHT(S)<br>
9 NINE(S)

Figurative constants may be used in the singular to denote the constant itself or in the plural to imply a string of constants.

EXPRESSIONS

The programmer combines words and symbols into procedure statements to direct computer operations. To facilitate the formulation of such statements showing the relationships and combinations of data names, conditional names, and constants, he has the assistance of arithmetic, relational, and logical expressions.

An arithmetic expression is a sequence of data names, numeric constants, and/or mathematical functions that are combined with symbols which represent arithmetic operations.

Operations and functions available to the programmer and their proper GECOM form are shown in Figure 15. They are listed in priority order, from highest to lowest. All of the listed functions are readily available as part of the GE-225 standard subroutine library and need not be generated during source program compilation or manually by the programmer. Previously-prepared subroutines materially reduce compilation time and programmer effort.

The natural priority of the table can be overridden by parentheses. Parentheses cause the evaluation to be performed from within the innermost set of

3110 S3110. ALTER. . . .

This statement sets SW3150 to proceed to S3155 the next time it is processed. SW3150 handles the group suppression of printing of DEPT~NO. When a new department is detected at 3080, it is necessary to print that department number from working storage, but immediately after, blanks are moved to that working storage field (part of the Detail Record) and the MOVE of blanks must be bypassed until the next new department is encountered.

3115 S3115. MOVE. . . .

This statement places the contents of the memory location assigned to hold the job ticket department number to the memory locations assigned to hold the last department number and the working storage department number. The LAST~DEPT is for comparison with the department of the current Job Ticket to determine a change of department at 3080, while the department of working storage is to provide the department number for the first printing of a detail record for a new department, and blanks afterward.

3120 MAN~COUNT=. . . .

This is an assignment statement that sets to zero the memory locations reserved for the named field.

3125 S3125. ADD. . . .

The man count memory location is increased by one.

3130 ADD. . . .

The two named fields are added and the result replaces the previous value of ACC~REG~HRS.

3135 ADD. . . .

The two named fields are added and the result replaces the previous value of ACC~OT~HRS.

3140 IF. . . .

The contents of the LINE~COUNT memory location are compared with the constant, 51. If they are equal, control transfers to procedure statement S3170; if they are not equal, the next statement in sequence is taken (3145). LINE~COUNT = 51 indicates that the last line of a printer page has been printed and a new page (and new headings) must be started.

3145 S3145. WRITE. . . .

The DETAIL RECORD, defined in Data Division statements 4150 through 4180, which includes DEPT, MAN~NBR, NAME, JOB~CODE, REG~HRS, and OT~HRS fields, is printed as a line by the high-speed printer.

3150 SW3150. GO TO. . . .

This is another program switch similar to SW3085 and SW3107. It governs whether the detail record print line contains an actual department number or blanks.

3155 S3155. MOVE. . . .

This statement replaces the contents of the working storage DEPT field with blanks.

3160 ALTER. . . .

This statement changes the object of the GO statement at SW3150 from S3155 to S3075 to bypass S3155 and 3160 until a new department is read.

3165 GO TO. . . .

This statement unconditionally transfers control to S3145.

3170 S3170. PERFORM. . . .

Like statement 3065, this sentence transfers control to the WPH SECTION beginning at 3005. Upon completion of this section, control automatically reverts to the next statement in sequence, 3175. This is used to head up a new page after the capacity of the preceding page has been filled by a department's records.

3175 GO TO. . . .

This statement unconditionally transfers control to S3145.

3180 S3180. ALTER. . . .

This statement changes the object of the GO statement at SW 3107 from S3110 to S3182, so that CLOSE will occur after the final summary card is punched.

3181 GO TO. . . .

This statement unconditionally transfers control to S3100 to compute the final summary card TOTAL~HRS.

3182 S3182. CLOSE. . . .

This statement terminates processings of the JOB~FILE and the SUMMARY~FILE. The card counts for the card reader and the card punch are printed out on the console typewriter.

| No. | A | B | Not-A | Not-B | A AND B | A OR B |
|---|---|---|---|---|---|---|
| 1. | True | True | False | False | True | True |
| 2. | True | False | False | True | False | True |
| 3. | False | True | True | False | False | True |
| 4. | False | False | True | True | False | False |

Figure 17. Logical Expression Truth Table

| No. | A | B | C | D |
|---|---|---|---|---|
| 1 | $A_1$ | $B_1$ | $C_1$ | $D_1$ |
| 2 | $A_2$ | $B_2$ | $C_2$ | $D_2$ |
| 3 | $A_3$ | $B_3$ | $C_3$ | $D_3$ |
| 4 | $A_4$ | $B_4$ | $C_4$ | $D_4$ |
| 5 | $A_5$ | $B_5$ | $C_5$ | $D_5$ |

Figure 18. Simple Two-Dimensional Table

Lists and tables of data can be stored within a data processing system for program reference also, permitting the programmer to instruct the program to perform "table look-up" operations. Such tables are stored in series within the system instead of in the grid-like manner illustrated above. The same table in the data processor might appear as a list, shown in Figure 19.

Even though the table data is stored as a long list, the programmer can still readily specify the required table data in essentially the same manner as a clerk would in instructing another clerk how to use the table first shown. The clerk would specify the table name, then the horizontal row and vertical column headings: TABLE 1, row 3, column C. The GECOM programmer does the same thing in a similar shorthand:

TABLE~1 (3, 3)
meaning TABLE~1, row 3, column 3.

Lists, tables, and matrices can all be represented in GECOM source programs and are referred to generically as arrays. A list is a one-dimensional array; a table, two-dimensional.

A three-dimensional array can be depicted graphically as a series of two-dimensional planes; as shown in Figure 20. Three-dimensional arrays could also be represented in storage as a series of sequential lists (one for each plane) like that described for the example above.

Arrays are assigned identifying names by the programmer. To identify array values, subscripts are used to specify rows, columns, and planes.

One-dimensional list = A(I)
Two-dimensional table = A(I,J)
Three-dimensional table = A(I,J,K)

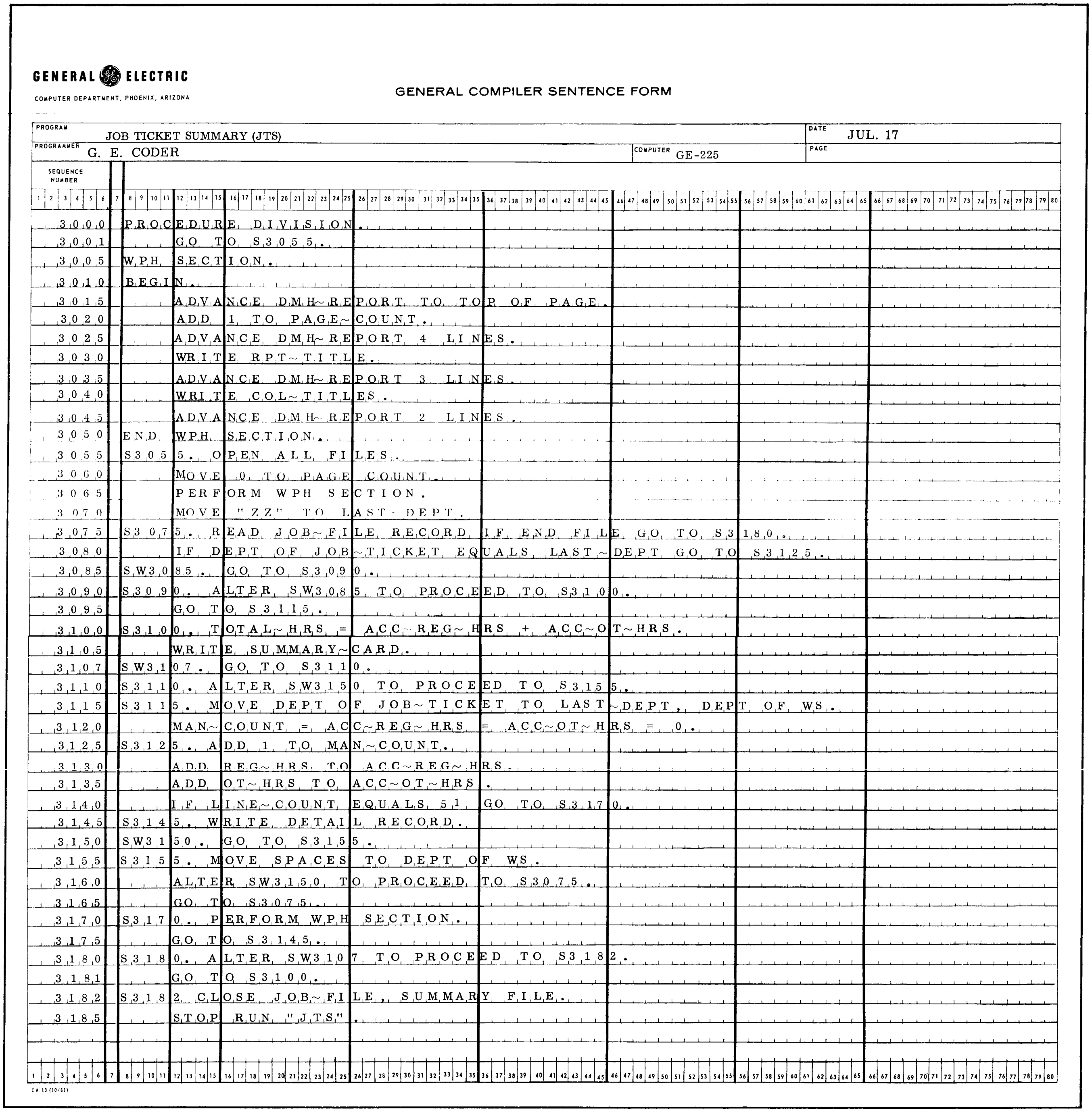
Subscripts can be written as arithmetic expressions, if need be, containing other subscripted arrays, and nested to up to ten deep in any one procedure statement.

LIST (A+C)
RATE (A-B*C, L(I,J),X)

In the second example A-B*C is the i-subscript, L(I,J) is the j-subscript, and X is the k-subscript for a matrix called RATE. Parentheses are always used to enclose subscripts which must immediately follow the array name.

1 $A_1$ $B_1$ $C_1$ $D_1$ 2 $A_2$ $B_2$ $C_2$ $D_2$ 3 $A_3$ $B_3$ $C_3$ $D_3$ 4 $A_4$ $B_4$ $C_4$ $D_4$ 5 $A_5$ $B_5$ $C_5$ $D_5$

Figure 19. A Two-Dimensional Table in Storage



GENERAL ELECTRIC
COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL COMPILER SENTENCE FORM

PROGRAM JOB TICKET SUMMARY (JTS)
DATE JUL. 17
PROGRAMMER G. E. CODER
COMPUTER GE-225
PAGE

SEQUENCE NUMBER

```
  3000 PROCEDURE DIVISION.
  3001     GO TO S3055.
  3005 WPH SECTION.
  3010 BEGIN.
  3015     ADVANCE DMH~REPORT TO TOP OF PAGE.
  3020     ADD 1 TO PAGE~COUNT.
  3025     ADVANCE DMH~REPORT 4 LINES.
  3030     WRITE RPT~TITLE.
  3035     ADVANCE DMH~REPORT 3 LINES.
  3040     WRITE COL~TITLES.
  3045     ADVANCE DMH~REPORT 2 LINES.
  3050 END WPH SECTION.
  3055 S3055. OPEN ALL FILES.
  3060     MOVE 0 TO PAGE COUNT.
  3065     PERFORM WPH SECTION.
  3070     MOVE "ZZ" TO LAST-DEPT.
  3075 S3075. READ JOB~FILE RECORD IF END FILE GO TO S3180.
  3080     IF DEPT OF JOB~TICKET EQUALS LAST~DEPT GO TO S3125.
  3085 SW3085. GO TO S3090.
  3090 S3090. ALTER SW3085 TO PROCEED TO S3100.
  3095     GO TO S3115.
  3100 S3100. TOTAL~HRS = ACC~REG~HRS + ACC~OT~HRS.
  3105     WRITE SUMMARY~CARD.
  3107 SW3107. GO TO S3110.
  3110 S3110. ALTER SW3150 TO PROCEED TO S3155.
  3115 S3115. MOVE DEPT OF JOB~TICKET TO LAST~DEPT, DEPT OF WS.
  3120     MAN~COUNT = ACC~REG~HRS = ACC~OT~HRS = 0.
  3125 S3125. ADD 1 TO MAN~COUNT.
  3130     ADD REG~HRS TO ACC~REG~HRS.
  3135     ADD OT~HRS TO ACC~OT~HRS.
  3140     IF LINE~COUNT EQUALS 51 GO TO S3170.
  3145 S3145. WRITE DETAIL RECORD.
  3150 SW3150. GO TO S3155.
  3155 S3155. MOVE SPACES TO DEPT OF WS.
  3160     ALTER SW3150 TO PROCEED TO S3075.
  3165     GO TO S3075.
  3170 S3170. PERFORM WPH SECTION.
  3175     GO TO S3145.
  3180 S3180. ALTER SW3107 TO PROCEED TO S3182.
  3181     GO TO S3100.
  3182 S3182 CLOSE JOB~FILE, SUMMARY FILE.
  3185     STOP RUN "JTS".
```

CA 13 (10/61)

Figure 34. Job Ticket Summary Procedure Division

# EXTENSIONS TO GECOM

## GECOM/REPORT WRITER

The GECOM/Report Writer requires the same compiling configuration as Basic GECOM, and is an extension of the basic compiler. Report writing programs can readily be described in the Basic GECOM language, but the Report Writer facilitates report preparation by enabling the user to describe reports concisely on a layout form which can be inserted into the GECOM Data Division. It also provides such features as automatic page and line control, facilitates programming, and provides better documentations of report writing programs.

Report specifications are written within the framework of a GECOM source program, and, in straightforward situations, are contained entirely within the Data and Environment Divisions. A knowledge of file and report formats and which record fields are the file sequence keys is all that is needed beyond a knowledge of GECOM to prepare procedure statements for most business reports. The user need only define the unique features of his job outside of the normal file processing procedure. The Report Writer tailors the basic framework to the programmer's needs and produces an object program for execution. The primary advantages to be gained by this method of description are minimized programming and debugging effort and readily-understandable program documentation.

With proper preparation of the source program, the Report Writer with GECOM will generate an object program which:

1. Prints report headings once at the beginning of the report.
2. Prints report footings once at the end of the report.
3. Maintains page control by line count and skips to a new page as specified.
4. Maintains line spacing on the page.
5. Prints page headings at the top of each report page.
6. Prints page footings at the bottom of each report page.
7. Numbers pages.
8. Issues detail lines according to the presence or absence of control conditions.
9. Accumulates detail field values to one or more levels of total.
10. Counts detail field conditions and detail lines to one or more levels of total.
11. Detects control breaks at one or more levels to control tabulation, issue control totals, and issue control headings.
12. Edits data fields for reporting by zero suppression, character insertion, fixing or floating dollar signs, and fixing or floating arithmetic signs.
13. Assigns and calculates values for report fields.
14. Reads a single file on one or more reels.
15. Reads successive files on multifile reels.
16. Performs normal file opening and closing functions.
17. Creates final totals and terminates reports at end of input.
18. Prepares a report(s) file for deferred printing.

Report descriptions are contained in the Report Section of the GECOM Data Division, under the heading REPORT SECTION, immediately following the File Section. All entries in this section must conform to the format of the Report Description Form, Figure 21, which is used in place of the standard GECOM Data Division form. Not shown are the supporting entries required in the Working Storage Section of the Data Division. Figure 21 illustrates a typical report as laid out in the Report Section of the Data Division, while Figure 22 shows the resulting printed report after processing of the object program containing the report description.

## GECOM/TABSOL

The GECOM/TABSOL extension requires the same compiling configuration as Basic GECOM and allows source programs to be described in tabular form. Although the same programs could be described in the basic GECOM procedural sentences, certain benefits are provided by the TABSOL extension.

TABSOL, which stands for Tabular Systems Oriented Language, is basically a structuring technique used to systematically describe the step by step decision logic in the process of solving a problem. The basic advantage of the TABSOL language is that it is easily learned and understood and can be applied to many analytical situations.

The I~O~CONTROL sentence is used only if non-standard label-checking rerun information and/or multifile magnetic tapes are required.

The FILE~CONTROL sentence is used when the source program requires the identification and/or assignment of input/output files or hardware units. If the source program does not process input/output data, the FILE~CONTROL sentence can be omitted.

The COMPUTATION~MODE sentence is used when it is desired to perform computations on data in floating point format using floating point arithmetic.

For the Job Ticket Summary problem, the Environment Division would be prepared as shown in Figure 33.

The General Compiler Sentence Form is used; heading information, such as program and programmer identification are discretionary. Actual line entries must adhere to the rules detailed in the GE-225 GECOM Language Specifications. Some of these rules are mentioned in the line entry explanations that follow.

2000 ENVIRONMENT DIVISION.

The division heading is always the first entry for the division. The heading should begin in column 8 (recommended) or may be indented any number of spaces to the right. The heading must be followed by a period and no other information should follow on that line.

2005 and 2010 OBJECT~ COMPUTER.

If this sentence is used, the sentence name should be started in column 8 and followed by a period. The sentence can start on the same line as the sentence name. In Figure 33, the compiler interprets the sentence to mean that the object program is to be performed on a GE-225 system with a 8192 word memory (2 MODULES) and the object program is to be input via card reader. To accomplish this, the General Compiler must produce the object program on punched cards via the card punch. Note that the sentence was too long to be completed on one line and was carried over to line 2010 and indented for clarity.

2015 FILE~CONTROL.

Like other sentence names, this one begins in column 8 as recommended. The first sentence is begun immediately after the name (with a blank between) and terminated with a period. All subsequent sentences must begin on a new line. The 2015 sentence in Figure 33 assigns the JOB ~ FILE (input) to the card reader buffer. The General Compiler interprets this to mean that data input through the card reader is to be treated as job file data.

2020 SELECT SUMMARY~FILE. . . .

This sentence assigns the SUMMARY~FILE to the card punch for output.

2025 SELECT DMH~REPORT. . . .

This sentence assigns the DMH REPORT to the high-speed printer for output. The DMH REPORT is considered as an output file and is therefore assigned to a peripheral like all files in the FILE~CONTROL Section.

PROCEDURE DIVISION PREPARATION

Once the programmer has flow charted the procedure to be followed and has defined all input and output data, it becomes relatively easy to state the processing steps to be followed in producing the desired output.

The programmer, having developed a working knowledge of GECOM language elements (verbs, names, constants, expressions, etc.) and their effects upon the object program, is prepared to document the procedure. Figure 34 illustrates the completed General Compiler Sentence Form for the Procedure Division of the Job Ticket Summary Problem. By relating the individual procedure statements and their explanations below to the flow charts in Figures 29 through 31, the overall procedure is more readily understood.

3000 PROCEDURE DIVISION.

Invariably the first entry for this division (and others) is the division name. It must be entered starting (preferably) in column 8 and terminated with a period.

3001 GO. . .

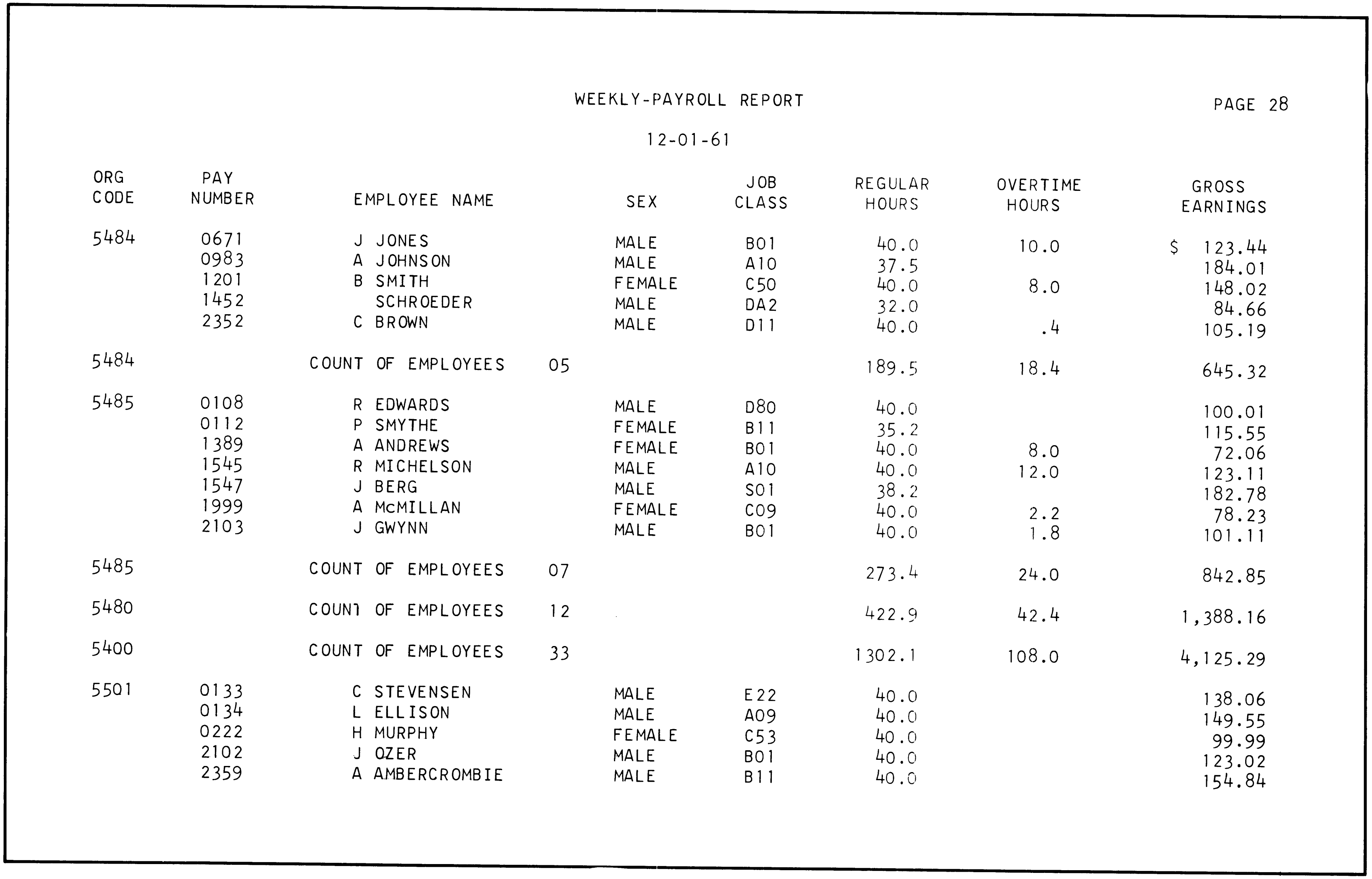
This opening sentence immediately and unconditionally transfers operation to the sentence identified by the sentence name, S3055.

3005 WPH SECTION.

This statement indicates that all procedure statements that follow are to be considered part of the WPH (Write Printer Heading) section until an END SECTION is encountered.

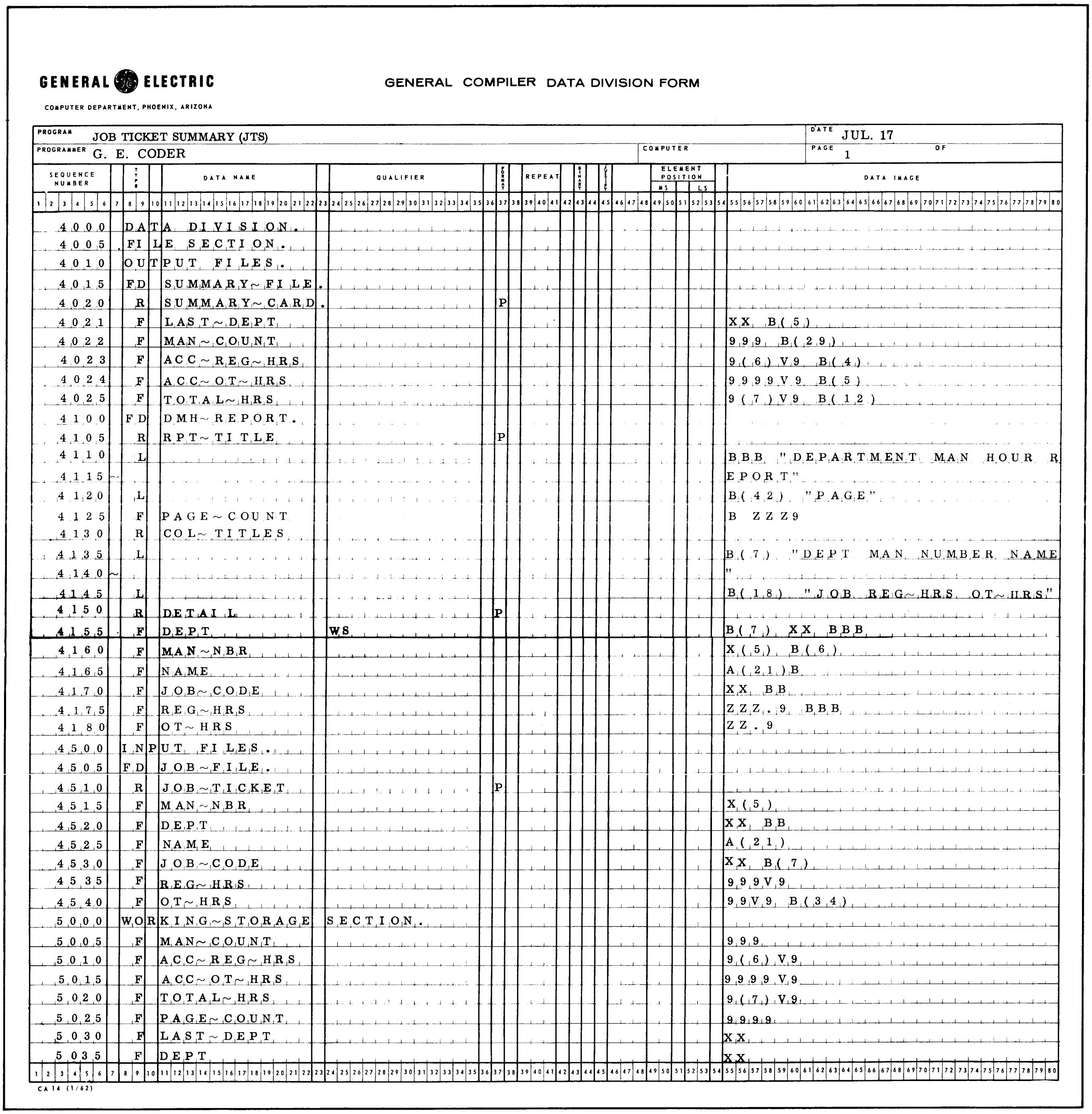
3010 through 3045

These statements comprise the WPH section which functions to advance the high-speed printer paper to the top of the page (3015), count pages (3020), space paper to the first print position (3025), print out the report title as defined by the literal entry at 4110 of the Data Division (3030), space paper to the next print line (3035), print out the column titles defined at 4135 through 4145 (3040),



```
                                     WEEKLY-PAYROLL REPORT                                      PAGE 28

                                           12-01-61

ORG      PAY                                                  JOB      REGULAR    OVERTIME       GROSS
CODE    NUMBER         EMPLOYEE NAME               SEX       CLASS      HOURS       HOURS       EARNINGS

5484     0671          J JONES                     MALE       B01        40.0        10.0      $  123.44
         0983          A JOHNSON                   MALE       A10        37.5                     184.01
         1201          B SMITH                     FEMALE     C50        40.0         8.0         148.02
         1452            SCHROEDER                 MALE       DA2        32.0                      84.66
         2352          C BROWN                     MALE       D11        40.0          .4         105.19

5484                 COUNT OF EMPLOYEES     05                          189.5        18.4         645.32

5485     0108          R EDWARDS                   MALE       D80        40.0                     100.01
         0112          P SMYTHE                    FEMALE     B11        35.2                     115.55
         1389          A ANDREWS                   FEMALE     B01        40.0         8.0          72.06
         1545          R MICHELSON                 MALE       A10        40.0        12.0         123.11
         1547          J BERG                      MALE       S01        38.2                     182.78
         1999          A McMILLAN                  FEMALE     C09        40.0         2.2          78.23
         2103          J GWYNN                     MALE       B01        40.0         1.8         101.11

5485                 COUNT OF EMPLOYEES     07                          273.4        24.0         842.85

5480                 COUNT OF EMPLOYEES     12                          422.9        42.4       1,388.16

5400                 COUNT OF EMPLOYEES     33                         1302.1       108.0       4,125.29

5501     0133          C STEVENSEN                 MALE       E22        40.0                     138.06
         0134          L ELLISON                   MALE       A09        40.0                     149.55
         0222          H MURPHY                    FEMALE     C53        40.0                      99.99
         2102          J OZER                      MALE       B01        40.0                     123.02
         2359          A AMBERCROMBIE              MALE       B11        40.0                     154.84
```

Figure 22. Report Writer Sample Report



GENERAL ELECTRIC

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL COMPILER DATA DIVISION FORM

PROGRAM JOB TICKET SUMMARY (JTS)

DATE JUL. 17

PROGRAMMER G. E. CODER

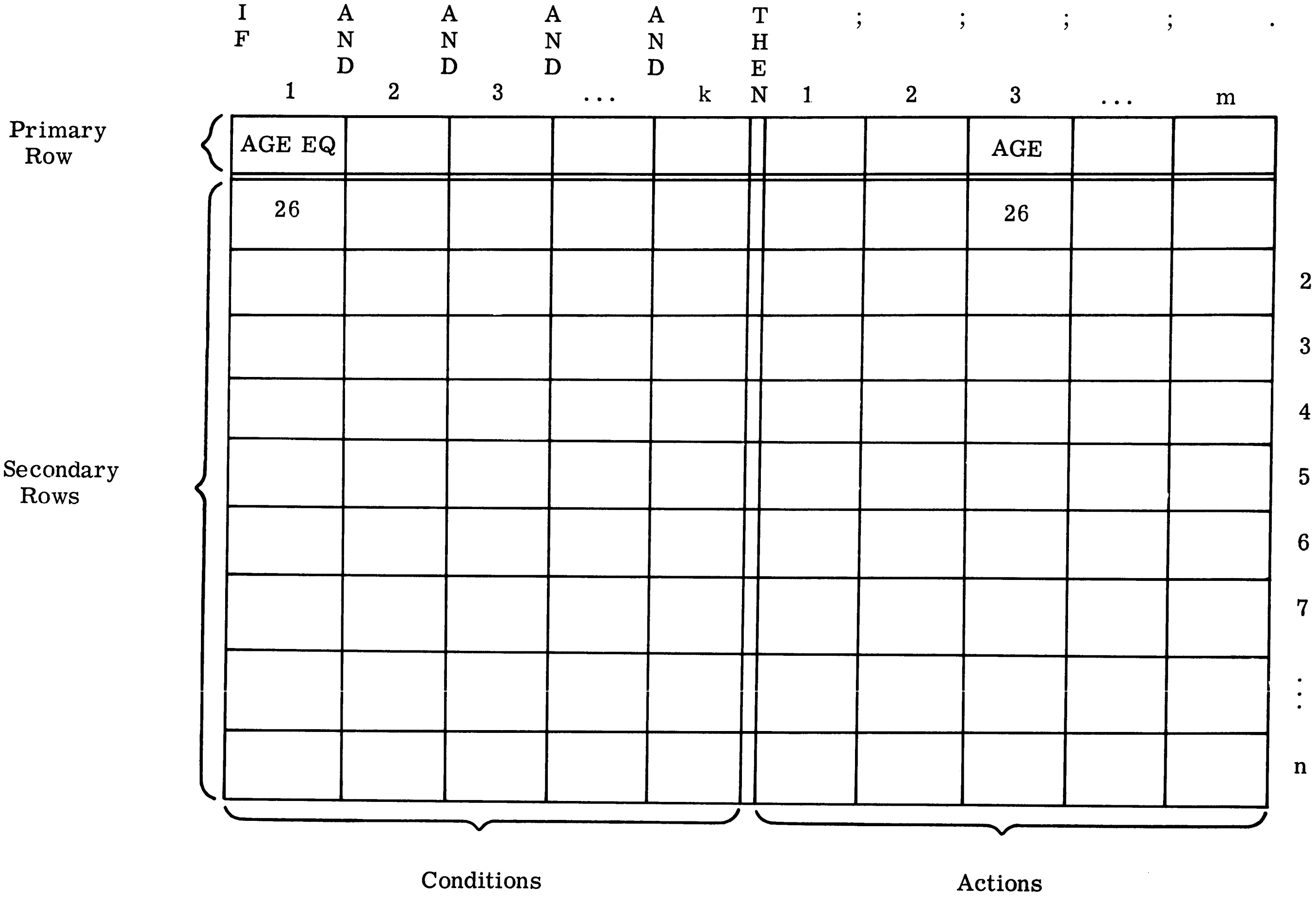
COMPUTER

PAGE 1 OF

| SEQUENCE NUMBER | | TYPE | DATA NAME | QUALIFIER | FORMAT | REPEAT | BINARY | JUSTIFY | ELEMENT POSITION MS | ELEMENT POSITION LS | DATA IMAGE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 4000 | | DATA DIVISION. | | | | | | | | | |
| 4005 | | FILE SECTION. | | | | | | | | | |
| 4010 | | OUTPUT FILES. | | | | | | | | | |
| 4015 | | FD | SUMMARY~FILE. | | | | | | | | |
| 4020 | | R | SUMMARY~CARD. | | P | | | | | | |
| 4021 | | F | LAST~DEPT | | | | | | | | XX B(5) |
| 4022 | | F | MAN~COUNT | | | | | | | | 999 B(29) |
| 4023 | | F | ACC~REG~HRS | | | | | | | | 9(6)V9 B(4) |
| 4024 | | F | ACC~OT~HRS | | | | | | | | 9999V9 B(5) |
| 4025 | | F | TOTAL~HRS | | | | | | | | 9(7)V9 B(12) |
| 4100 | | FD | DMH~REPORT. | | | | | | | | |
| 4105 | | R | RPT~TITLE | | P | | | | | | |
| 4110 | | L | | | | | | | | | BBB "DEPARTMENT MAN HOUR R |
| 4115 | ~ | | | | | | | | | | EPORT" |
| 4120 | | L | | | | | | | | | B(42) "PAGE" |
| 4125 | | F | PAGE~COUNT | | | | | | | | B ZZZ9 |
| 4130 | | R | COL~TITLES | | | | | | | | |
| 4135 | | L | | | | | | | | | B(7) "DEPT MAN NUMBER NAME |
| 4140 | ~ | | | | | | | | | | " |
| 4145 | | L | | | | | | | | | B(18) "JOB REG~HRS OT~HRS" |
| 4150 | | R | DETAIL | | P | | | | | | |
| 4155 | | F | DEPT | WS | | | | | | | B(7) XX BBB |
| 4160 | | F | MAN~NBR | | | | | | | | X(5) B(6) |
| 4165 | | F | NAME | | | | | | | | A(21)B |
| 4170 | | F | JOB~CODE | | | | | | | | XX BB |
| 4175 | | F | REG~HRS | | | | | | | | ZZZ.9 BBB |
| 4180 | | F | OT~HRS | | | | | | | | ZZ.9 |
| 4500 | | INPUT FILES. | | | | | | | | | |
| 4505 | | FD | JOB~FILE. | | | | | | | | |
| 4510 | | R | JOB~TICKET | | P | | | | | | |
| 4515 | | F | MAN~NBR | | | | | | | | X(5) |
| 4520 | | F | DEPT | | | | | | | | XX BB |
| 4525 | | F | NAME | | | | | | | | A(21) |
| 4530 | | F | JOB~CODE | | | | | | | | XX B(7) |
| 4535 | | F | REG~HRS | | | | | | | | 999V9 |
| 4540 | | F | OT~HRS | | | | | | | | 99V9 B(34) |
| 5000 | | WORKING~STORAGE | | SECTION. | | | | | | | |
| 5005 | | F | MAN~COUNT | | | | | | | | 999 |
| 5010 | | F | ACC~REG~HRS | | | | | | | | 9(6)V9 |
| 5015 | | F | ACC~OT~HRS | | | | | | | | 9999V9 |
| 5020 | | F | TOTAL~HRS | | | | | | | | 9(7)V9 |
| 5025 | | F | PAGE~COUNT | | | | | | | | 9999 |
| 5030 | | F | LAST~DEPT | | | | | | | | XX |
| 5035 | | F | DEPT | | | | | | | | XX |

CA 14 (1/62)

Figure 32. Job Ticket Summary Data Division



| | 1 | 2 | 3 | ... | k | 1 | 2 | 3 | ... | m | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Primary Row | AGE EQ | | | | | | | AGE | | | |
| Secondary Rows | 26 | | | | | | | 26 | | | |
| | | | | | | | | | | | 2 |
| | | | | | | | | | | | 3 |
| | | | | | | | | | | | 4 |
| | | | | | | | | | | | 5 |
| | | | | | | | | | | | 6 |
| | | | | | | | | | | | 7 |
| | | | | | | | | | | | ⋮ |
| | | | | | | | | | | | n |

**Figure 23. Decision Table Format**

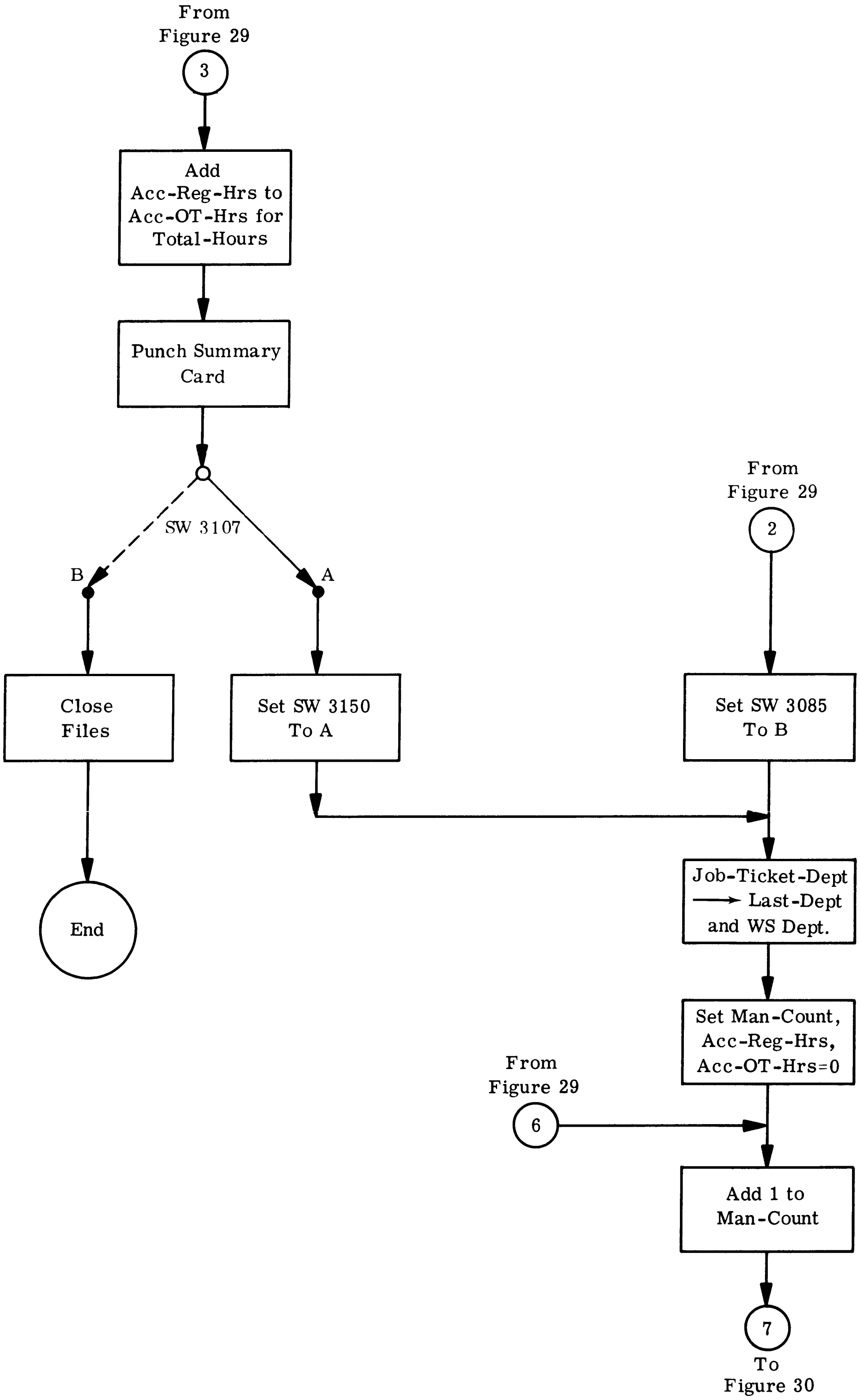
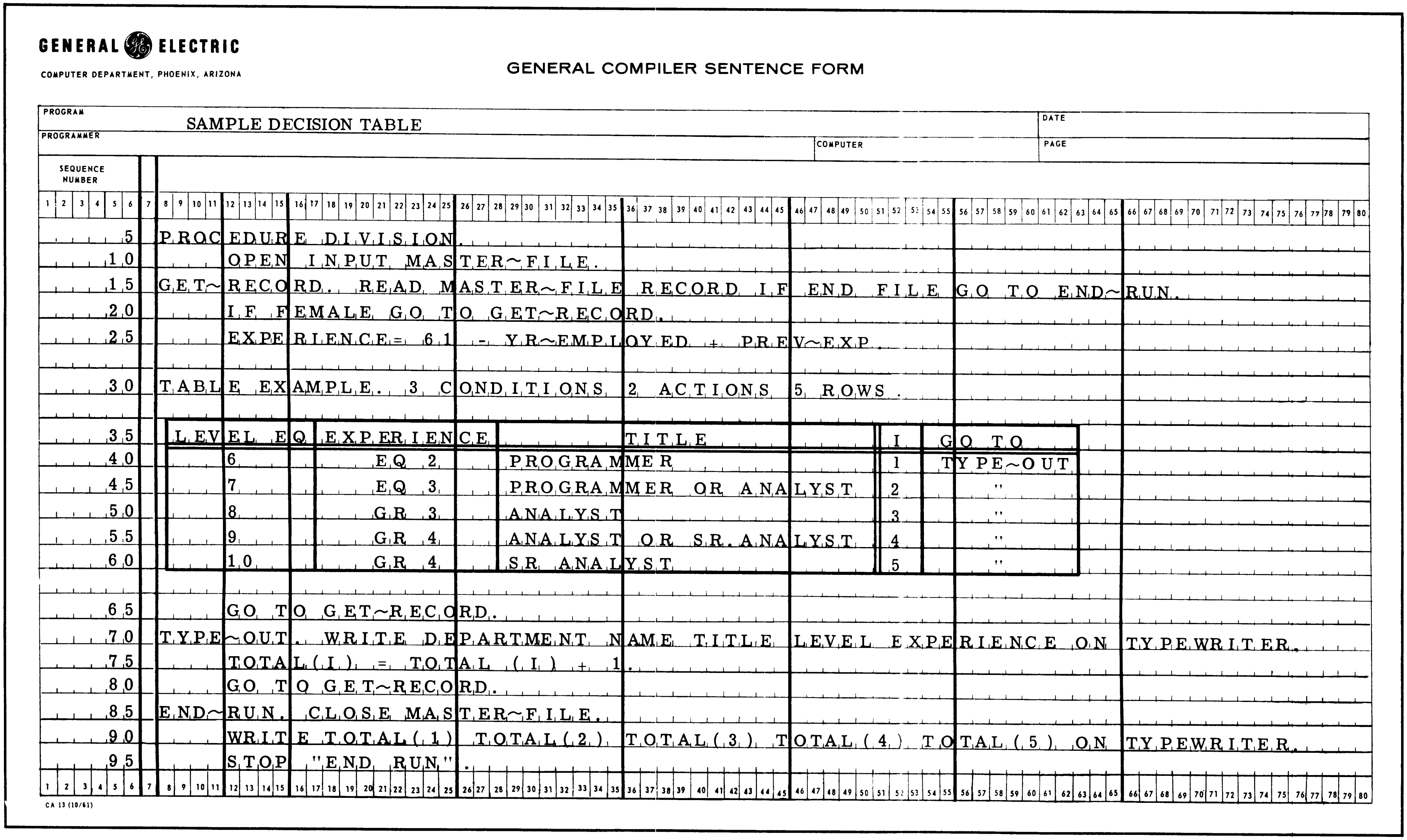


**Figure 31. Job Ticket Summary Flow Chart (continued)**



GENERAL ELECTRIC

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL COMPILER SENTENCE FORM

PROGRAM SAMPLE DECISION TABLE

PROGRAMMER

DATE

COMPUTER

PAGE

SEQUENCE NUMBER

```
     5 PROCEDURE DIVISION.
    10     OPEN INPUT MASTER~FILE.
    15 GET~RECORD. READ MASTER~FILE RECORD IF END FILE GO TO END~RUN.
    20     IF FEMALE GO TO GET~RECORD.
    25     EXPERIENCE = 61 - YR~EMPLOYED + PREV~EXP.

    30 TABLE EXAMPLE. 3 CONDITIONS 2 ACTIONS 5 ROWS.

    35  LEVEL EQ    EXPERIENCE      TITLE           I  GO TO
    40     6         EQ 2  PROGRAMMER               1  TYPE~OUT
    45     7         EQ 3  PROGRAMMER OR ANALYST    2    "
    50     8         GR 3  ANALYST                  3    "
    55     9         GR 4  ANALYST OR SR. ANALYST   4    "
    60     10        GR 4  SR ANALYST               5    "

    65     GO TO GET~RECORD.
    70 TYPE~OUT. WRITE DEPARTMENT NAME TITLE LEVEL EXPERIENCE ON TYPEWRITER.
    75     TOTAL(I) = TOTAL (I) + 1.
    80     GO TO GET~RECORD.
    85 END~RUN. CLOSE MASTER~FILE.
    90     WRITE TOTAL(1) TOTAL(2) TOTAL(3) TOTAL(4) TOTAL(5) ON TYPEWRITER.
    95     STOP "END RUN".
```

CA 13 (10/61)

Figure 24. Sample TABSOL Table in GECOM

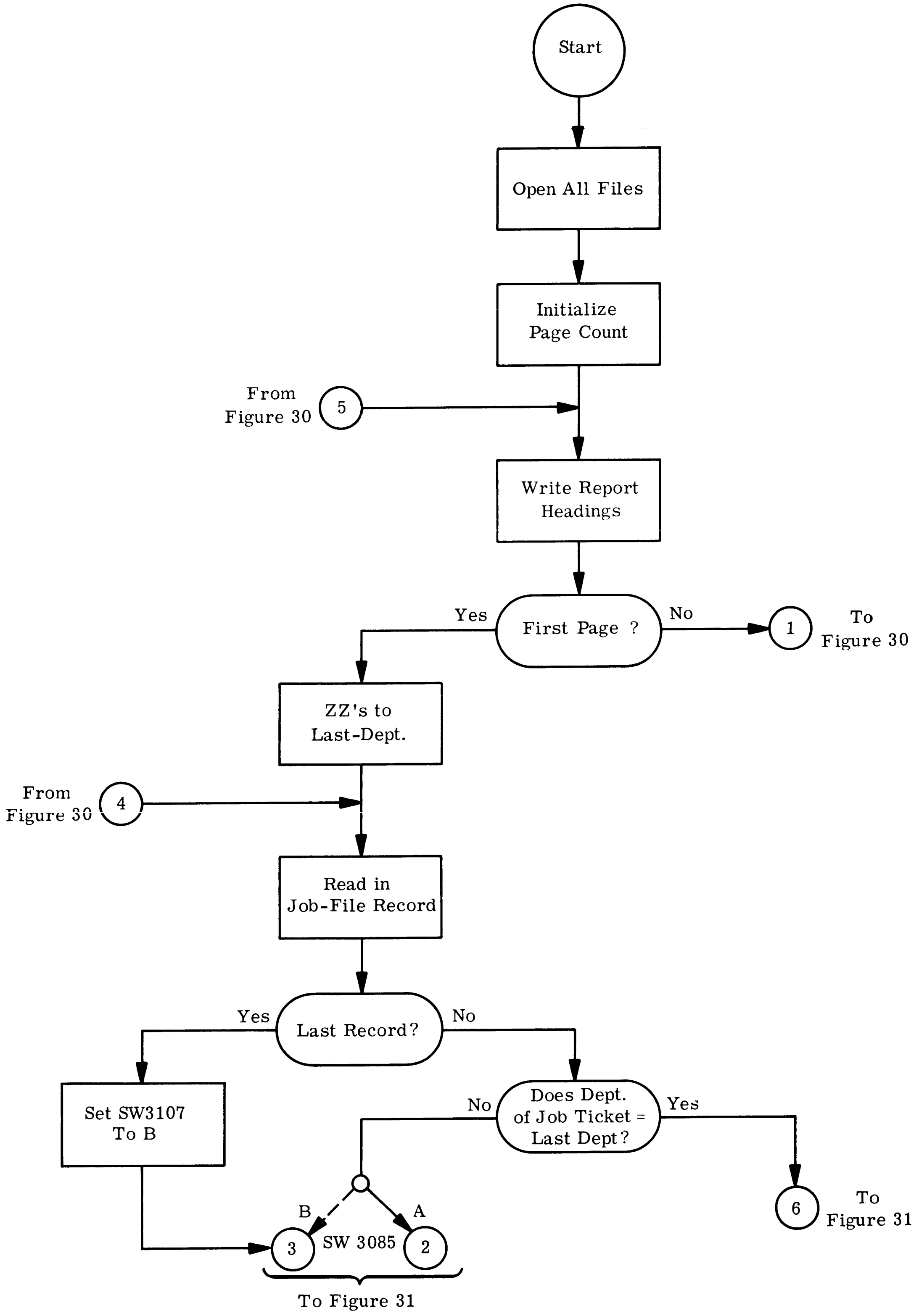


Figure 29. Job Ticket Summary Flow Chart

# APPLICATION OF BASIC GECOM

## GENERAL

To more closely relate the use of the GECOM system to actual applications, the following pages carry a sample problem through the programming process. Although not all of the capabilities of Basic GECOM are exercised, enough material is presented to provide perspective and insight into the scope of GECOM.

First, the problem is presented and the objective is defined.

Second, the procedure to be followed is outlined, the required inputs and desired outputs are identified, and a flow chart is prepared.

Third, the source program is produced. Each of the four divisions of the GECOM source program are illustrated and discussed where appropriate. The compilations and debugging of the object program, performed on the GE-225, are not covered in detail. Procedures for compilation are fully discussed in the GE-225 GECOM Operations Manual, CD 225H1.

Finally, the outputs of the compilation process, the Edited List and the object program, are presented and discussed.

## DEFINING THE PROBLEM

The sample problem selected involves a typical manufacturing plant that uses job ticket records for each employee to produce time and job accounting data. Assuming that the individual Job Ticket Records follow the format illustrated in Figure 25, the problem is to prepare a program that will produce two outputs:

1. A punched card summary record for each department, showing the:

   Department Number
   Number of Men
   Accumulated Regular Hours
   Accuumulated Overtime Hours
   Total Hours

2. A printed report providing, by department and man number, this information for each man:

   Department Number
   Man Number
   Name
   Job
   Regular Hours
   Overtime Hours

Figure 26 shows a representative punched card summary record, while Figure 27 shows the desired printed report.

In an actual application, it is quite possible that the input data (the Job Ticket Record) and the desired outputs (the Job Ticket Summary and the Department Man Hour Report) would not already be defined. The problem might be as informally stated as, "we need to know what our people are doing and how long it takes to do it."

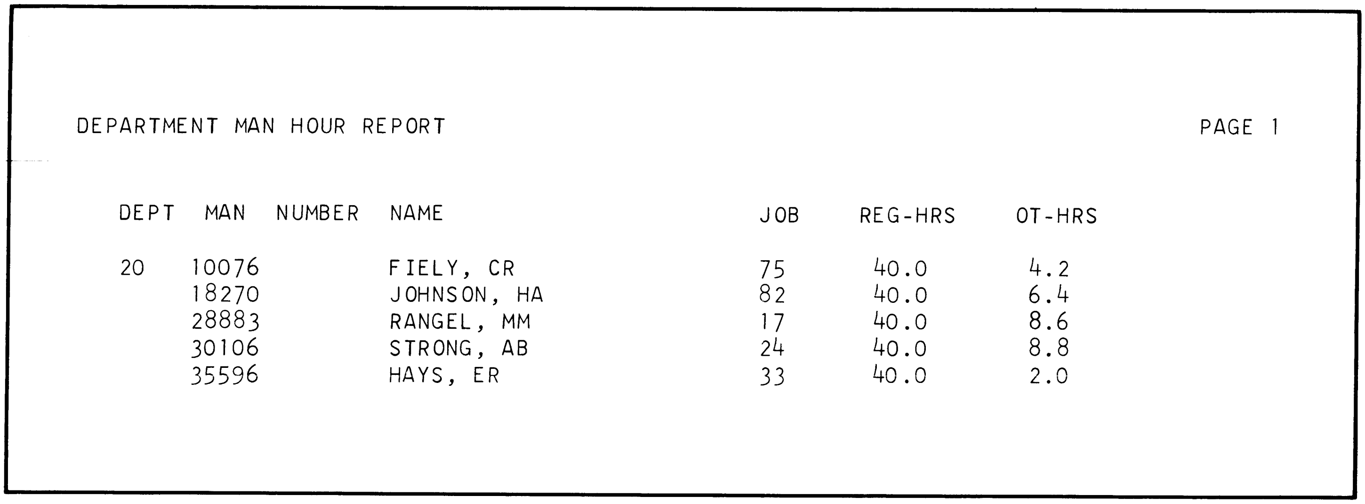
In these circumstances, the problem would also entail determining what input data is needed, how to collect it, and how to record it for computer input. It would also be necessary to determine (more precisely than the quoted problem states) what output is desired and what form and organization it should follow.

Here, these preliminary decisions have been made. It remains for the programmer to document the process to be performed by the data processor, detail the procedure the program must follow (via a flow chart), and prepare the source program.

## PLOTTING THE SOLUTION

In the sample problem, documenting the process involves little more than translating the problem statement into a diagram. The input is already defined; the purpose of the program has been stated; and the desired outputs have been described. Graphically the process chart appears as shown in Figure 28.

A more realistic application might involve several inputs and outputs via several media. Additionally, multiple "runs" or processes by the data processor



```
DEPARTMENT MAN HOUR REPORT                                          PAGE 1

  DEPT   MAN  NUMBER   NAME                      JOB     REG-HRS     OT-HRS

  20    10076          FIELY, CR                 75       40.0        4.2
        18270          JOHNSON, HA               82       40.0        6.4
        28883          RANGEL, MM                17       40.0        8.6
        30106          STRONG, AB                24       40.0        8.8
        35596          HAYS, ER                  33       40.0        2.0
```

Figure 27. Department Man Hour Report

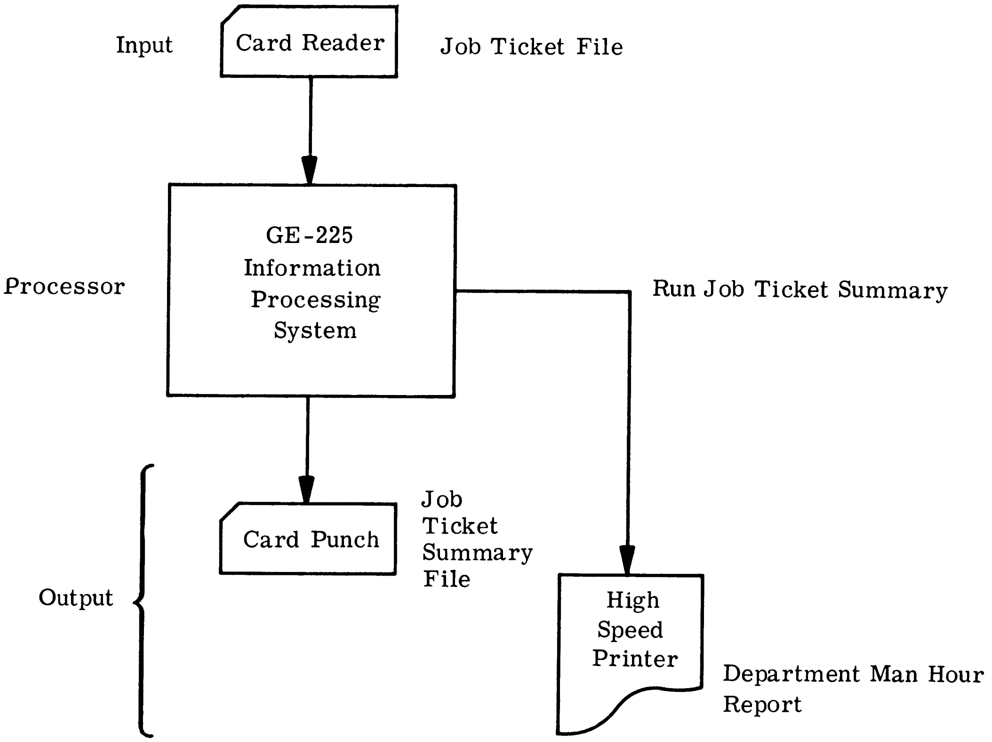
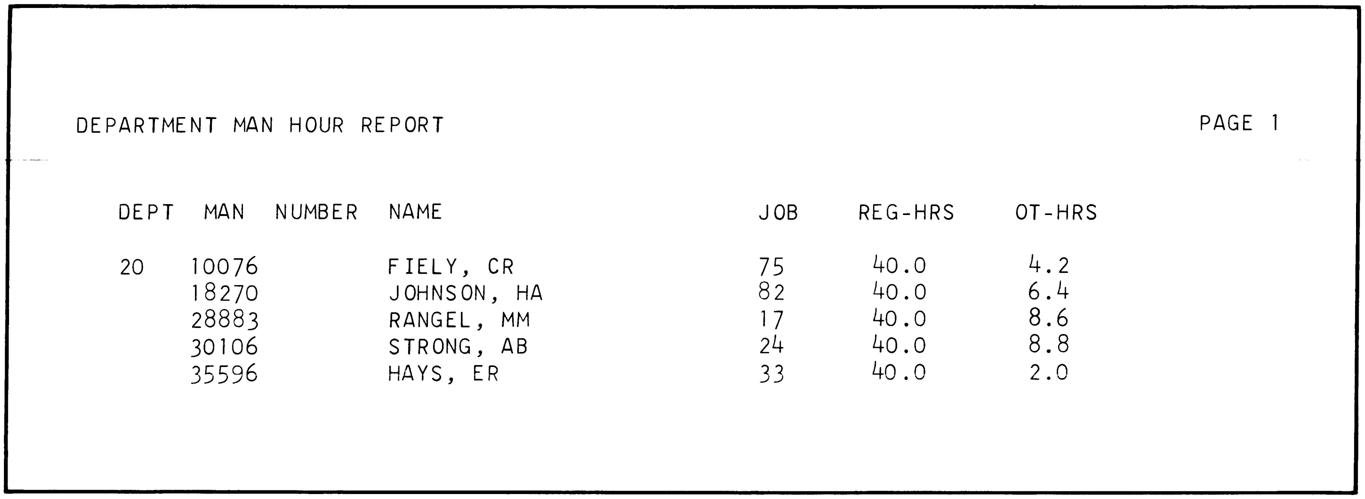


Figure 28. Process Chart for Job Ticket Summary



```
DEPARTMENT MAN HOUR REPORT                                              PAGE 1

  DEPT   MAN  NUMBER  NAME                          JOB     REG-HRS     OT-HRS

  20    10076         FIELY, CR                     75       40.0        4.2
        18270         JOHNSON, HA                   82       40.0        6.4
        28883         RANGEL, MM                    17       40.0        8.6
        30106         STRONG, AB                    24       40.0        8.8
        35596         HAYS, ER                      33       40.0        2.0
```

Figure 27. Department Man Hour Report

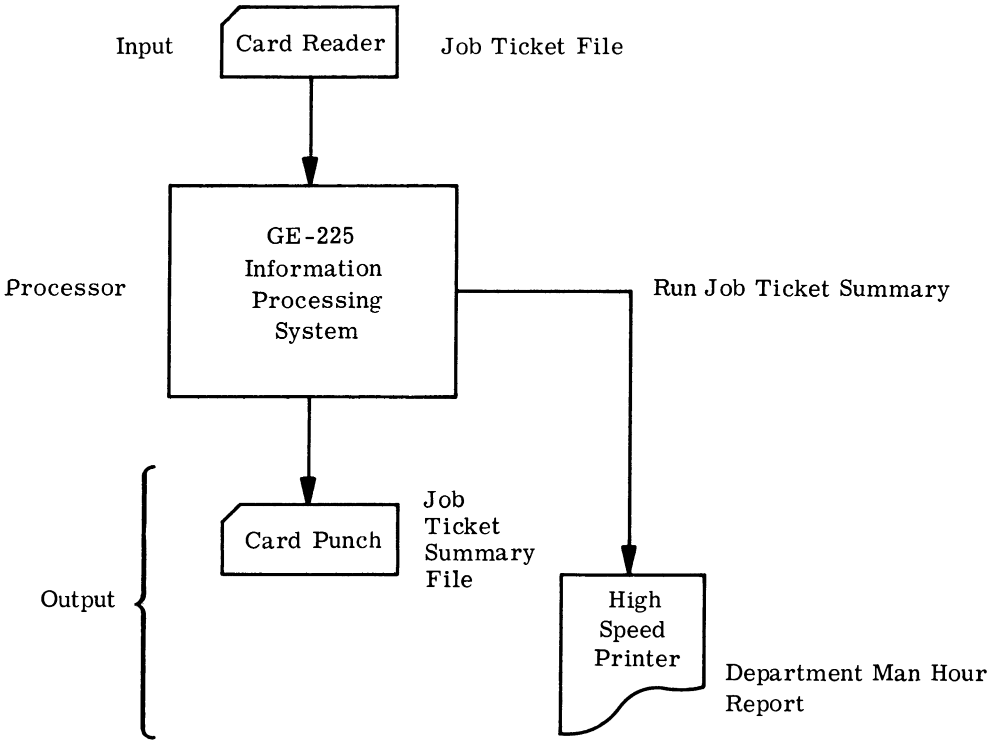


Figure 28. Process Chart for Job Ticket Summary

# APPLICATION OF BASIC GECOM

## GENERAL

To more closely relate the use of the GECOM system to actual applications, the following pages carry a sample problem through the programming process. Although not all of the capabilities of Basic GECOM are exercised, enough material is presented to provide perspective and insight into the scope of GECOM.

First, the problem is presented and the objective is defined.

Second, the procedure to be followed is outlined, the required inputs and desired outputs are identified, and a flow chart is prepared.

Third, the source program is produced. Each of the four divisions of the GECOM source program are illustrated and discussed where appropriate. The compilations and debugging of the object program, performed on the GE-225, are not covered in detail. Procedures for compilation are fully discussed in the GE-225 GECOM Operations Manual, CD 225H1.

Finally, the outputs of the compilation process, the Edited List and the object program, are presented and discussed.

## DEFINING THE PROBLEM

The sample problem selected involves a typical manufacturing plant that uses job ticket records for each employee to produce time and job accounting data. Assuming that the individual Job Ticket Records follow the format illustrated in Figure 25, the problem is to prepare a program that will produce two outputs:

1. A punched card summary record for each department, showing the:

   Department Number
   Number of Men
   Accumulated Regular Hours
   Accuumulated Overtime Hours
   Total Hours

2. A printed report providing, by department and man number, this information for each man:

   Department Number
   Man Number
   Name
   Job
   Regular Hours
   Overtime Hours

Figure 26 shows a representative punched card summary record, while Figure 27 shows the desired printed report.

In an actual application, it is quite possible that the input data (the Job Ticket Record) and the desired outputs (the Job Ticket Summary and the Department Man Hour Report) would not already be defined. The problem might be as informally stated as, "we need to know what our people are doing and how long it takes to do it."

In these circumstances, the problem would also entail determining what input data is needed, how to collect it, and how to record it for computer input. It would also be necessary to determine (more precisely than the quoted problem states) what output is desired and what form and organization it should follow.

Here, these preliminary decisions have been made. It remains for the programmer to document the process to be performed by the data processor, detail the procedure the program must follow (via a flow chart), and prepare the source program.

## PLOTTING THE SOLUTION

In the sample problem, documenting the process involves little more than translating the problem statement into a diagram. The input is already defined; the purpose of the program has been stated; and the desired outputs have been described. Graphically the process chart appears as shown in Figure 28.

A more realistic application might involve several inputs and outputs via several media. Additionally, multiple "runs" or processes by the data processor

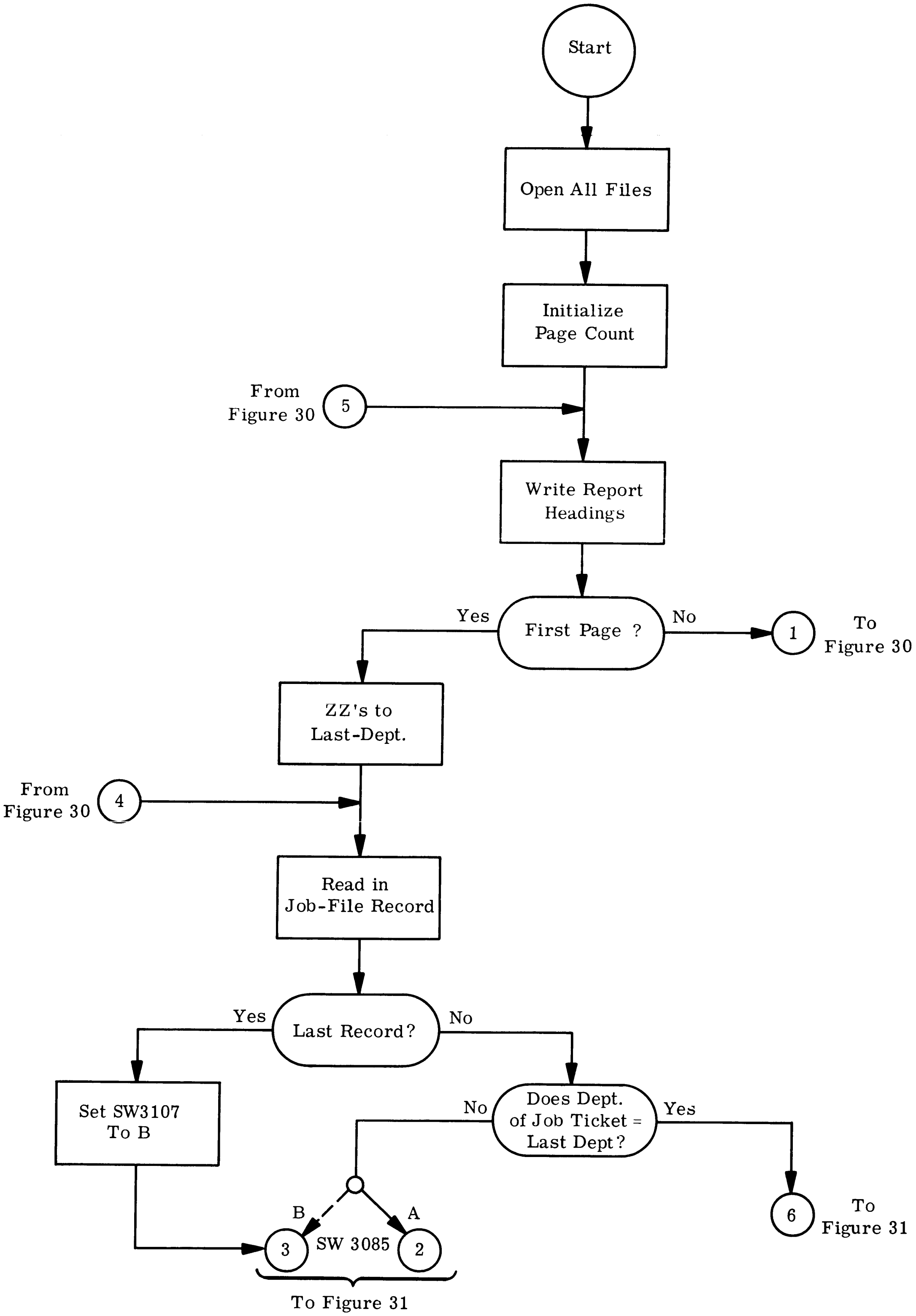
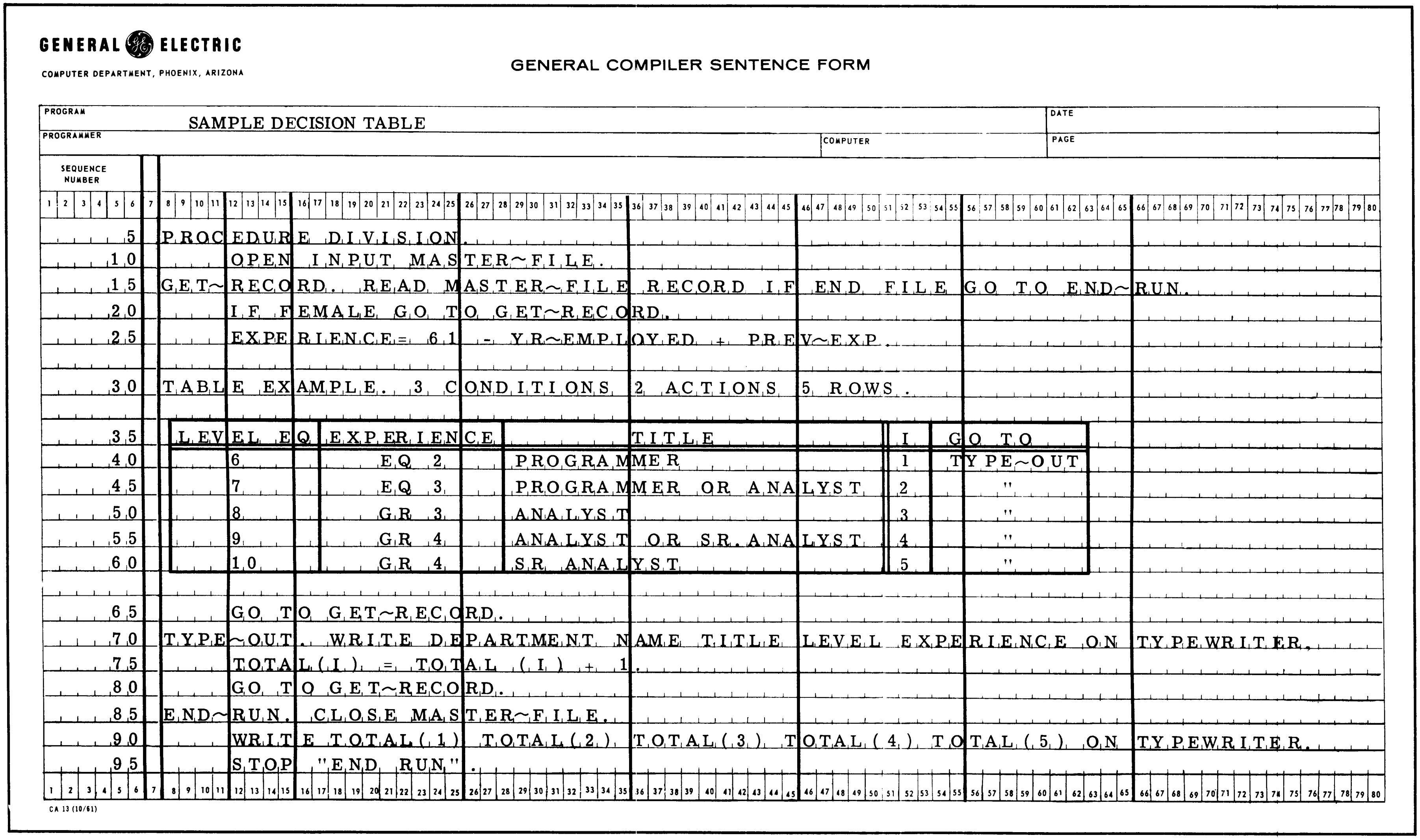


Figure 29. Job Ticket Summary Flow Chart



GENERAL ELECTRIC
COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL COMPILER SENTENCE FORM

PROGRAM SAMPLE DECISION TABLE
PROGRAMMER
COMPUTER
DATE
PAGE

SEQUENCE NUMBER

```
     5  PROCEDURE DIVISION.
    10      OPEN INPUT MASTER~FILE.
    15  GET~RECORD. READ MASTER~FILE RECORD IF END FILE GO TO END~RUN.
    20      IF FEMALE GO TO GET~RECORD.
    25      EXPERIENCE = 61 - YR~EMPLOYED + PREV~EXP.

    30  TABLE EXAMPLE. 3 CONDITIONS 2 ACTIONS 5 ROWS.

    35   LEVEL EQ  EXPERIENCE             TITLE                 I   GO TO
    40          6          EQ 2   PROGRAMMER                    1   TYPE~OUT
    45          7          EQ 3   PROGRAMMER OR ANALYST         2     "
    50          8          GR 3   ANALYST                       3     "
    55          9          GR 4   ANALYST OR SR. ANALYST        4     "
    60          10         GR 4   SR ANALYST                    5     "

    65      GO TO GET~RECORD.
    70  TYPE~OUT. WRITE DEPARTMENT NAME TITLE LEVEL EXPERIENCE ON TYPEWRITER.
    75      TOTAL(I) = TOTAL (I) + 1.
    80      GO TO GET~RECORD.
    85  END~RUN. CLOSE MASTER~FILE.
    90      WRITE TOTAL(1) TOTAL(2) TOTAL(3) TOTAL(4) TOTAL(5) ON TYPEWRITER.
    95      STOP "END RUN".
```

CA 13 (10/61)

Figure 24. Sample TABSOL Table in GECOM

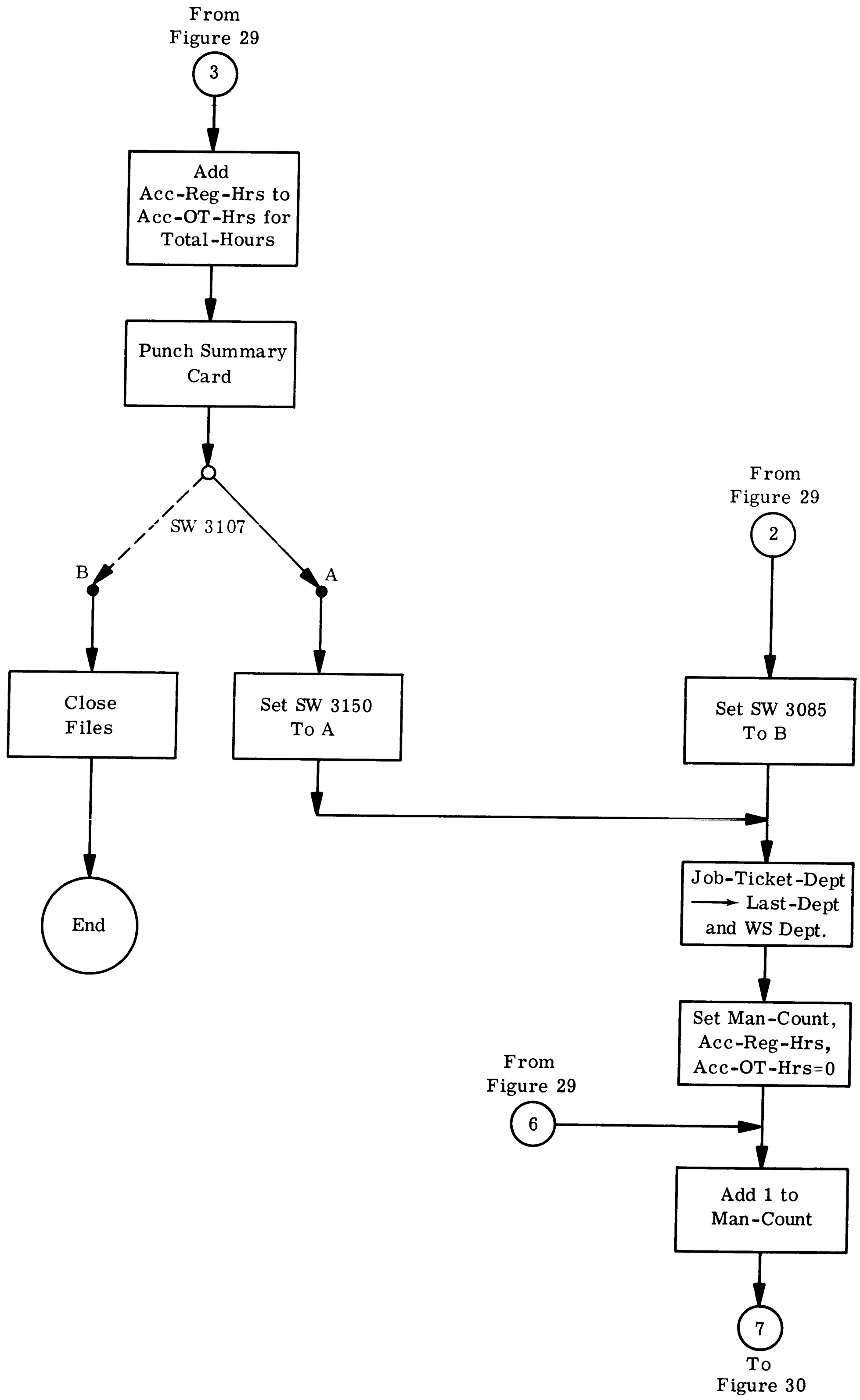


Figure 31. Job Ticket Summary Flow Chart (continued)

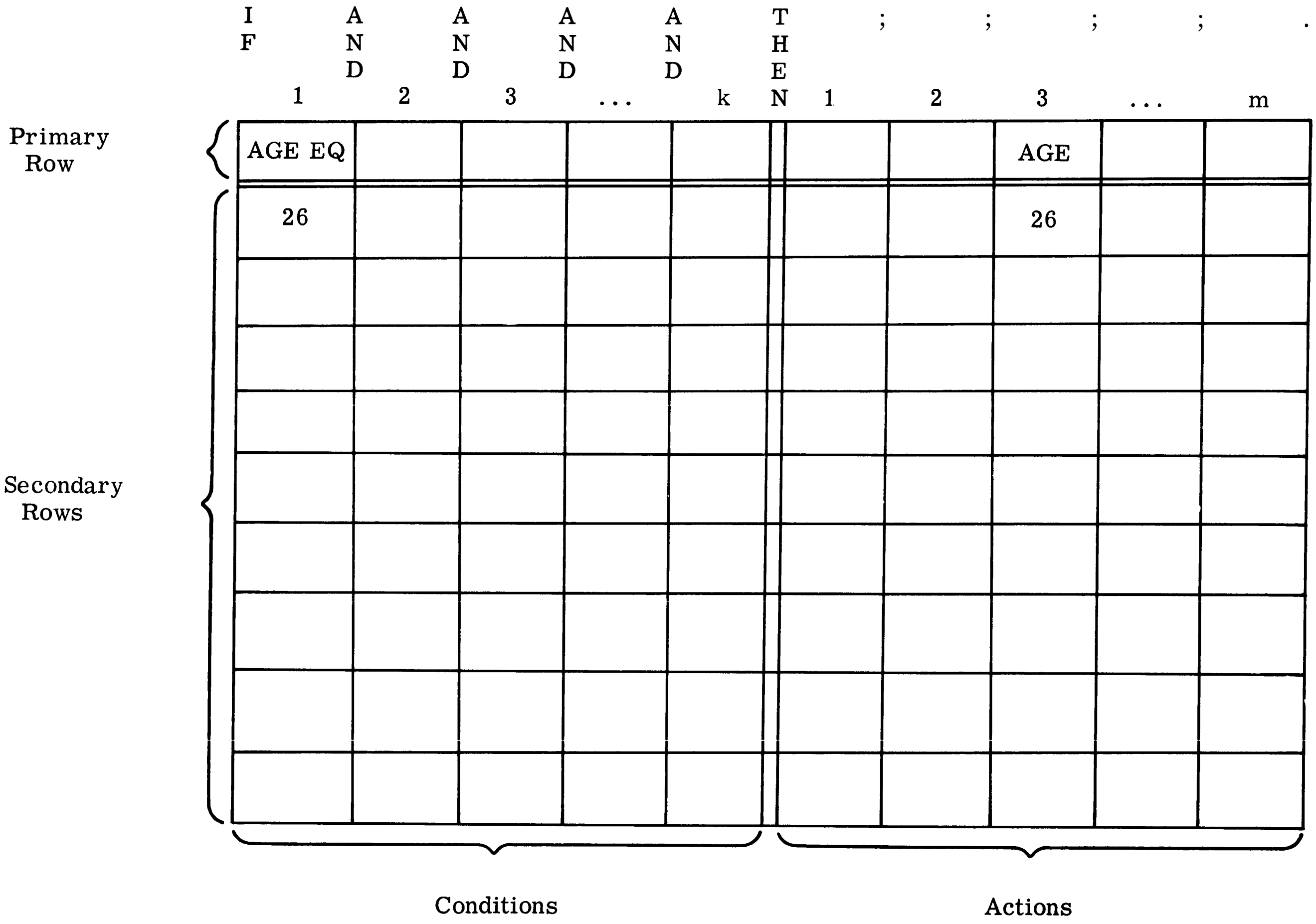
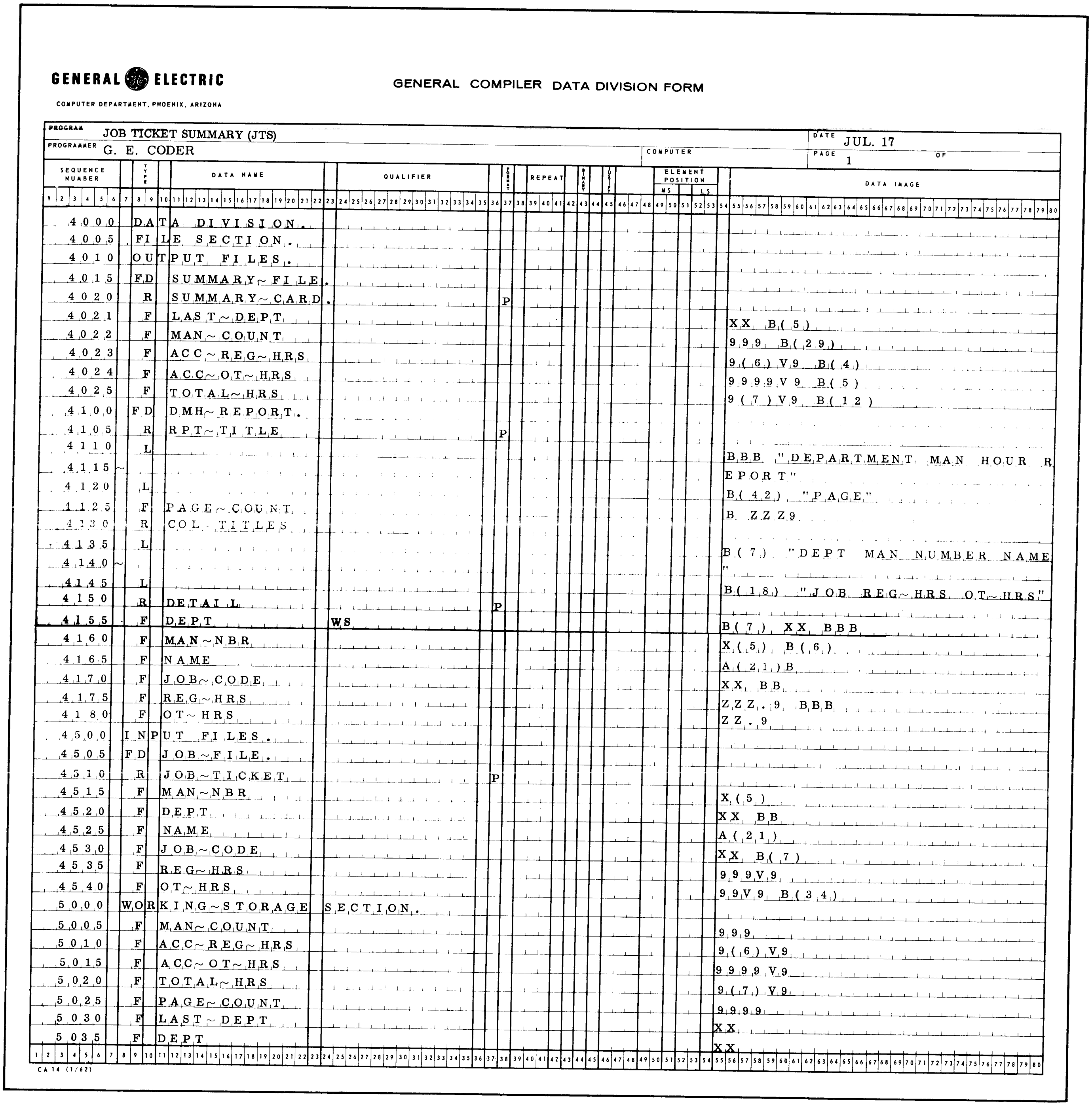


**Figure 23. Decision Table Format**



GENERAL ELECTRIC

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL COMPILER DATA DIVISION FORM

PROGRAM JOB TICKET SUMMARY (JTS)

PROGRAMMER G. E. CODER

COMPUTER

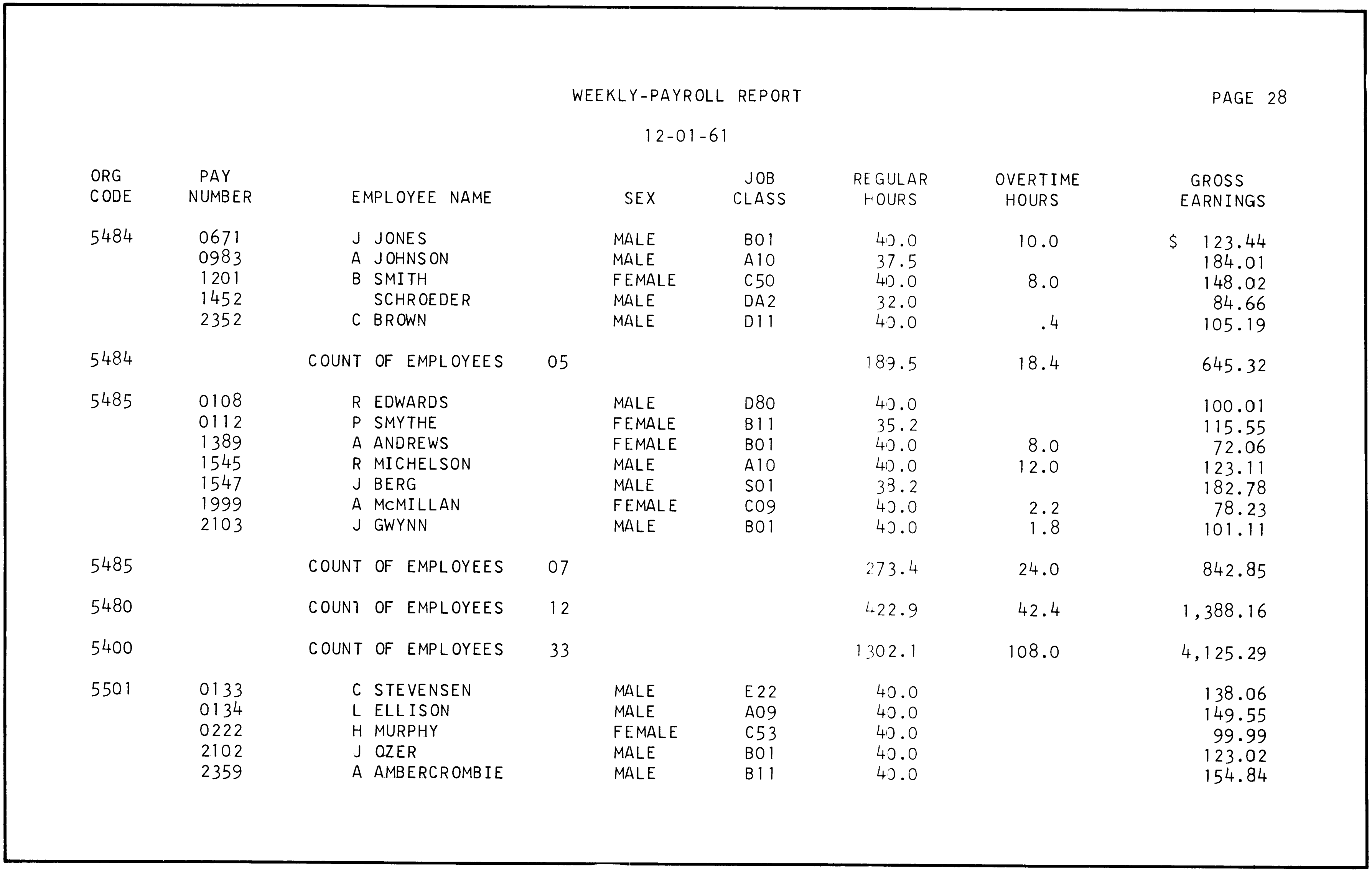
DATE JUL. 17

PAGE 1 OF

| SEQUENCE NUMBER | | TYPE | | DATA NAME | | QUALIFIER | FORMAT | REPEAT | BINARY | JUSTIFY | ELEMENT POSITION MS | ELEMENT POSITION LS | DATA IMAGE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4000 | | DA | T | A DIVISION. | | | | | | | | | |
| 4005 | | FI | L | E SECTION. | | | | | | | | | |
| 4010 | | OU | T | PUT FILES. | | | | | | | | | |
| 4015 | | FD | | SUMMARY~FILE | . | | | | | | | | |
| 4020 | | R | | SUMMARY~CARD | . | | P | | | | | | |
| 4021 | | F | | LAST~DEPT | | | | | | | | | XX B(5) |
| 4022 | | F | | MAN~COUNT | | | | | | | | | 999 B(29) |
| 4023 | | F | | ACC~REG~HRS | | | | | | | | | 9(6)V9 B(4) |
| 4024 | | F | | ACC~OT~HRS | | | | | | | | | 9999V9 B(5) |
| 4025 | | F | | TOTAL~HRS | | | | | | | | | 9(7)V9 B(12) |
| 4100 | | FD | | DMH~REPORT. | | | | | | | | | |
| 4105 | | R | | RPT~TITLE | | | P | | | | | | |
| 4110 | | L | | | | | | | | | | | BBB "DEPARTMENT MAN HOUR R |
| 4115 | ~ | | | | | | | | | | | | EPORT" |
| 4120 | | L | | | | | | | | | | | B(42) "PAGE" |
| 4125 | | F | | PAGE~COUNT | | | | | | | | | B ZZZ9 |
| 4130 | | R | | COL-TITLES | | | | | | | | | |
| 4135 | | L | | | | | | | | | | | B(7) "DEPT MAN NUMBER NAME |
| 4140 | ~ | | | | | | | | | | | | " |
| 4145 | | L | | | | | | | | | | | B(18) "JOB REG~HRS OT~HRS" |
| 4150 | | R | | DETAIL | | | P | | | | | | |
| 4155 | | F | | DEPT | | WS | | | | | | | B(7) XX BBB |
| 4160 | | F | | MAN~NBR | | | | | | | | | X(5) B(6) |
| 4165 | | F | | NAME | | | | | | | | | A(21)B |
| 4170 | | F | | JOB~CODE | | | | | | | | | XX BB |
| 4175 | | F | | REG~HRS | | | | | | | | | ZZZ.9 BBB |
| 4180 | | F | | OT~HRS | | | | | | | | | ZZ.9 |
| 4500 | | IN | P | UT FILES. | | | | | | | | | |
| 4505 | | FD | | JOB~FILE. | | | | | | | | | |
| 4510 | | R | | JOB~TICKET | | | P | | | | | | |
| 4515 | | F | | MAN~NBR | | | | | | | | | X(5) |
| 4520 | | F | | DEPT | | | | | | | | | XX BB |
| 4525 | | F | | NAME | | | | | | | | | A(21) |
| 4530 | | F | | JOB~CODE | | | | | | | | | XX B(7) |
| 4535 | | F | | REG~HRS | | | | | | | | | 999V9 |
| 4540 | | F | | OT~HRS | | | | | | | | | 99V9 B(34) |
| 5000 | | WO | R | KING~STORAGE | | SECTION. | | | | | | | |
| 5005 | | F | | MAN~COUNT | | | | | | | | | 999 |
| 5010 | | F | | ACC~REG~HRS | | | | | | | | | 9(6)V9 |
| 5015 | | F | | ACC~OT~HRS | | | | | | | | | 9999V9 |
| 5020 | | F | | TOTAL~HRS | | | | | | | | | 9(7)V9 |
| 5025 | | F | | PAGE~COUNT | | | | | | | | | 9999 |
| 5030 | | F | | LAST~DEPT | | | | | | | | | XX |
| 5035 | | F | | DEPT | | | | | | | | | XX |

CA 14 (1/62)

Figure 32. Job Ticket Summary Data Division



```
                                   WEEKLY-PAYROLL REPORT                                PAGE 28

                                         12-01-61

ORG     PAY                                              JOB     REGULAR    OVERTIME        GROSS
CODE    NUMBER         EMPLOYEE NAME            SEX      CLASS   HOURS      HOURS        EARNINGS

5484     0671          J JONES                  MALE     B01       40.0       10.0       $  123.44
         0983          A JOHNSON                MALE     A10       37.5                     184.01
         1201          B SMITH                  FEMALE   C50       40.0        8.0          148.02
         1452            SCHROEDER              MALE     DA2       32.0                      84.66
         2352          C BROWN                  MALE     D11       40.0         .4          105.19

5484                COUNT OF EMPLOYEES   05                       189.5       18.4          645.32

5485     0108          R EDWARDS                MALE     D80       40.0                     100.01
         0112          P SMYTHE                 FEMALE   B11       35.2                     115.55
         1389          A ANDREWS                FEMALE   B01       40.0        8.0           72.06
         1545          R MICHELSON              MALE     A10       40.0       12.0          123.11
         1547          J BERG                   MALE     S01       33.2                     182.78
         1999          A McMILLAN               FEMALE   C09       40.0        2.2           78.23
         2103          J GWYNN                  MALE     B01       40.0        1.8          101.11

5485                COUNT OF EMPLOYEES   07                       273.4       24.0          842.85

5480                COUN1 OF EMPLOYEES   12                       422.9       42.4        1,388.16

5400                COUNT OF EMPLOYEES   33                      1302.1      108.0        4,125.29

5501     0133          C STEVENSEN              MALE     E22       40.0                     138.06
         0134          L ELLISON                MALE     A09       40.0                     149.55
         0222          H MURPHY                 FEMALE   C53       40.0                      99.99
         2102          J OZER                   MALE     B01       40.0                     123.02
         2359          A AMBERCROMBIE           MALE     B11       40.0                     154.84
```

Figure 22. Report Writer Sample Report

The I~O~CONTROL sentence is used only if nonstandard label-checking rerun information and/or multifile magnetic tapes are required.

The FILE~CONTROL sentence is used when the source program requires the identification and/or assignment of input/output files or hardware units. If the source program does not process input/output data, the FILE~CONTROL sentence can be omitted.

The COMPUTATION~MODE sentence is used when it is desired to perform computations on data in floating point format using floating point arithmetic.

For the Job Ticket Summary problem, the Environment Division would be prepared as shown in Figure 33.

The General Compiler Sentence Form is used; heading information, such as program and programmer identification are discretionary. Actual line entries must adhere to the rules detailed in the GE-225 GECOM Language Specifications. Some of these rules are mentioned in the line entry explanations that follow.

2000 ENVIRONMENT DIVISION.

The division heading is always the first entry for the division. The heading should begin in column 8 (recommended) or may be indented any number of spaces to the right. The heading must be followed by a period and no other information should follow on that line.

2005 and 2010 OBJECT~ COMPUTER.

If this sentence is used, the sentence name should be started in column 8 and followed by a period. The sentence can start on the same line as the sentence name. In Figure 33, the compiler interprets the sentence to mean that the object program is to be performed on a GE-225 system with a 8192 word memory (2 MODULES) and the object program is to be input via card reader. To accomplish this, the General Compiler must produce the object program on punched cards via the card punch. Note that the sentence was too long to be completed on one line and was carried over to line 2010 and indented for clarity.

2015 FILE~CONTROL.

Like other sentence names, this one begins in column 8 as recommended. The first sentence is begun immediately after the name (with a blank between) and terminated with a period. All subsequent sentences must begin on a new line. The 2015 sentence in Figure 33 assigns the JOB ~ FILE (input) to the card reader buffer. The General Compiler interprets this to mean that data input through the card reader is to be treated as job file data.

2020 SELECT SUMMARY~FILE. . . .

This sentence assigns the SUMMARY~FILE to the card punch for output.

2025 SELECT DMH~REPORT. . . .

This sentence assigns the DMH REPORT to the high-speed printer for output. The DMH REPORT is considered as an output file and is therefore assigned to a peripheral like all files in the FILE~CONTROL Section.

PROCEDURE DIVISION PREPARATION

Once the programmer has flow charted the procedure to be followed and has defined all input and output data, it becomes relatively easy to state the processing steps to be followed in producing the desired output.

The programmer, having developed a working knowledge of GECOM language elements (verbs, names, constants, expressions, etc.) and their effects upon the object program, is prepared to document the procedure. Figure 34 illustrates the completed General Compiler Sentence Form for the Procedure Division of the Job Ticket Summary Problem. By relating the individual procedure statements and their explanations below to the flow charts in Figures 29 through 31, the overall procedure is more readily understood.

3000 PROCEDURE DIVISION.

Invariably the first entry for this division (and others) is the division name. It must be entered starting (preferably) in column 8 and terminated with a period.

3001 GO. . .

This opening sentence immediately and unconditionally transfers operation to the sentence identified by the sentence name, S3055.

3005 WPH SECTION.

This statement indicates that all procedure statements that follow are to be considered part of the WPH (Write Printer Heading) section until an END SECTION is encountered.

3010 through 3045

These statements comprise the WPH section which functions to advance the high-speed printer paper to the top of the page (3015), count pages (3020), space paper to the first print position (3025), print out the report title as defined by the literal entry at 4110 of the Data Division (3030), space paper to the next print line (3035), print out the column titles defined at 4135 through 4145 (3040),

# EXTENSIONS TO GECOM

## GECOM/REPORT WRITER

The GECOM/Report Writer requires the same compiling configuration as Basic GECOM, and is an extension of the basic compiler. Report writing programs can readily be described in the Basic GECOM language, but the Report Writer facilitates report preparation by enabling the user to describe reports concisely on a layout form which can be inserted into the GECOM Data Division. It also provides such features as automatic page and line control, facilitates programming, and provides better documentations of report writing programs.

Report specifications are written within the framework of a GECOM source program, and, in straightforward situations, are contained entirely within the Data and Environment Divisions. A knowledge of file and report formats and which record fields are the file sequence keys is all that is needed beyond a knowledge of GECOM to prepare procedure statements for most business reports. The user need only define the unique features of his job outside of the normal file processing procedure. The Report Writer tailors the basic framework to the programmer's needs and produces an object program for execution. The primary advantages to be gained by this method of description are minimized programming and debugging effort and readily-understandable program documentation.

With proper preparation of the source program, the Report Writer with GECOM will generate an object program which:

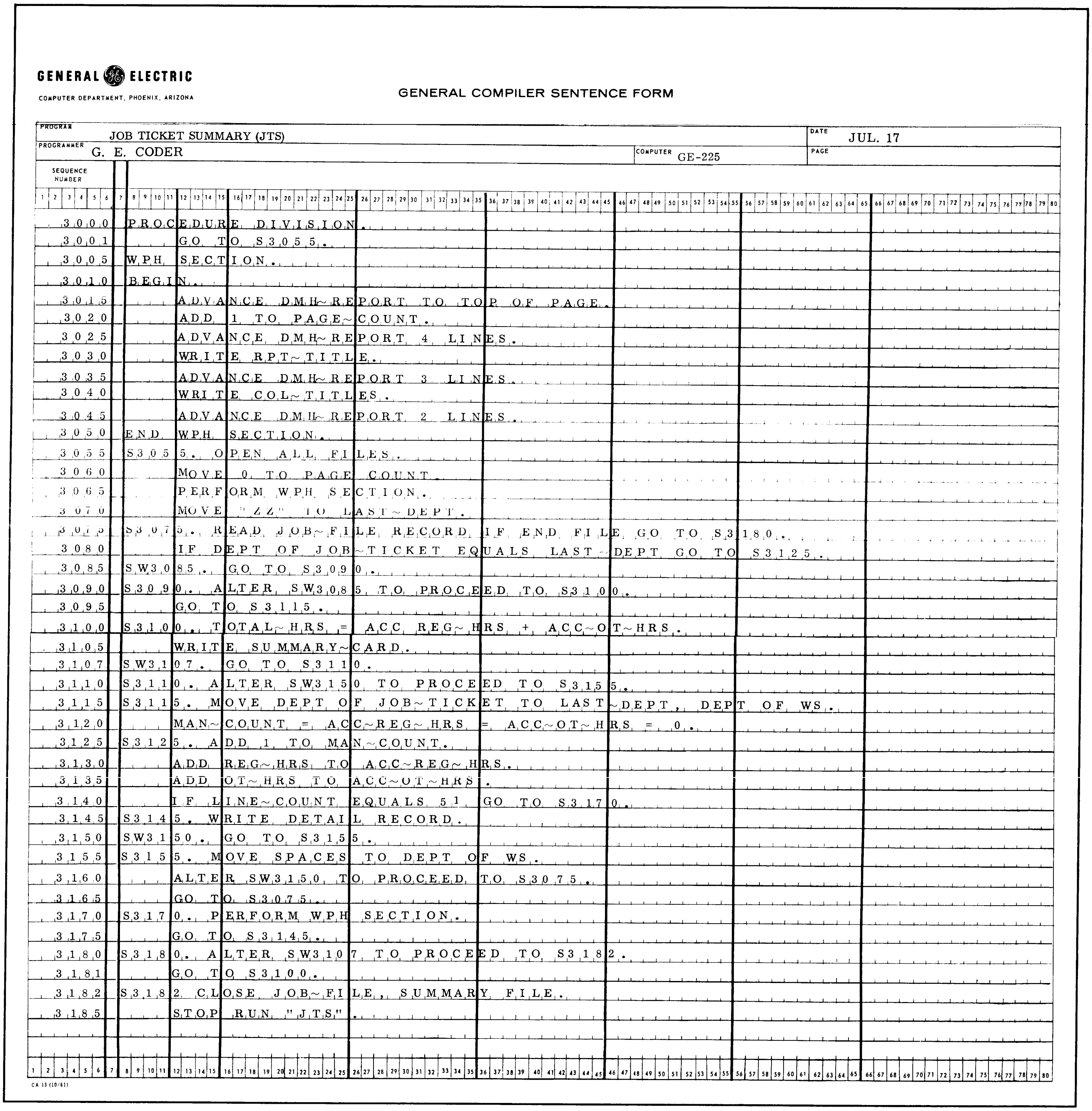
1. Prints report headings once at the beginning of the report.
2. Prints report footings once at the end of the report.
3. Maintains page control by line count and skips to a new page as specified.
4. Maintains line spacing on the page.
5. Prints page headings at the top of each report page.
6. Prints page footings at the bottom of each report page.
7. Numbers pages.
8. Issues detail lines according to the presence or absence of control conditions.
9. Accumulates detail field values to one or more levels of total.
10. Counts detail field conditions and detail lines to one or more levels of total.
11. Detects control breaks at one or more levels to control tabulation, issue control totals, and issue control headings.
12. Edits data fields for reporting by zero suppression, character insertion, fixing or floating dollar signs, and fixing or floating arithmetic signs.
13. Assigns and calculates values for report fields.
14. Reads a single file on one or more reels.
15. Reads successive files on multifile reels.
16. Performs normal file opening and closing functions.
17. Creates final totals and terminates reports at end of input.
18. Prepares a report(s) file for deferred printing.

Report descriptions are contained in the Report Section of the GECOM Data Division, under the heading REPORT SECTION, immediately following the File Section. All entries in this section must conform to the format of the Report Description Form, Figure 21, which is used in place of the standard GECOM Data Division form. Not shown are the supporting entries required in the Working Storage Section of the Data Division. Figure 21 illustrates a typical report as laid out in the Report Section of the Data Division, while Figure 22 shows the resulting printed report after processing of the object program containing the report description.

## GECOM/TABSOL

The GECOM/TABSOL extension requires the same compiling configuration as Basic GECOM and allows source programs to be described in tabular form. Although the same programs could be described in the basic GECOM procedural sentences, certain benefits are provided by the TABSOL extension.

TABSOL, which stands for Tabular Systems Oriented Language, is basically a structuring technique used to systematically describe the step by step decision logic in the process of solving a problem. The basic advantage of the TABSOL language is that it is easily learned and understood and can be applied to many analytical situations.



GENERAL ELECTRIC
COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL COMPILER SENTENCE FORM

| | | | |
|---|---|---|---|
| PROGRAM JOB TICKET SUMMARY (JTS) | | | DATE JUL. 17 |
| PROGRAMMER G. E. CODER | | COMPUTER GE-225 | PAGE |

```
  3000 PROCEDURE DIVISION.
  3001     GO TO S3055.
  3005 WPH SECTION.
  3010 BEGIN.
  3015     ADVANCE DMH~REPORT TO TOP OF PAGE.
  3020     ADD 1 TO PAGE~COUNT.
  3025     ADVANCE DMH~REPORT 4 LINES.
  3030     WRITE RPT~TITLE.
  3035     ADVANCE DMH~REPORT 3 LINES.
  3040     WRITE COL~TITLES.
  3045     ADVANCE DMH~REPORT 2 LINES.
  3050 END WPH SECTION.
  3055 S3055. OPEN ALL FILES.
  3060     MOVE 0 TO PAGE COUNT.
  3065     PERFORM WPH SECTION.
  3070     MOVE "ZZ" TO LAST~DEPT.
  3075 S3075. READ JOB~FILE RECORD IF END FILE GO TO S3180.
  3080     IF DEPT OF JOB~TICKET EQUALS LAST~DEPT GO TO S3125.
  3085 SW3085. GO TO S3090.
  3090 S3090. ALTER SW3085 TO PROCEED TO S3100.
  3095     GO TO S3115.
  3100 S3100. TOTAL~HRS = ACC REG~HRS + ACC~OT~HRS.
  3105     WRITE SUMMARY~CARD.
  3107 SW3107. GO TO S3110.
  3110 S3110. ALTER SW3150 TO PROCEED TO S3155.
  3115 S3115. MOVE DEPT OF JOB~TICKET TO LAST~DEPT, DEPT OF WS.
  3120     MAN~COUNT = ACC~REG~HRS = ACC~OT~HRS = 0.
  3125 S3125. ADD 1 TO MAN~COUNT.
  3130     ADD REG~HRS TO ACC~REG~HRS.
  3135     ADD OT~HRS TO ACC~OT~HRS.
  3140     IF LINE~COUNT EQUALS 51 GO TO S3170.
  3145 S3145. WRITE DETAIL RECORD.
  3150 SW3150. GO TO S3155.
  3155 S3155. MOVE SPACES TO DEPT OF WS.
  3160     ALTER SW3150 TO PROCEED TO S3075.
  3165     GO TO S3075.
  3170 S3170. PERFORM WPH SECTION.
  3175     GO TO S3145.
  3180 S3180. ALTER SW3107 TO PROCEED TO S3182.
  3181     GO TO S3100.
  3182 S3182 CLOSE JOB~FILE, SUMMARY FILE.
  3185     STOP RUN "JTS".
```

CA 13 (10/61)

Figure 34. Job Ticket Summary Procedure Division

| No. | A | B | Not-A | Not-B | A AND B | A OR B |
|---|---|---|---|---|---|---|
| 1. | True | True | False | False | True | True |
| 2. | True | False | False | True | False | True |
| 3. | False | True | True | False | False | True |
| 4. | False | False | True | True | False | False |

Figure 17. Logical Expression Truth Table

| No. | A | B | C | D |
|---|---|---|---|---|
| 1 | $A_1$ | $B_1$ | $C_1$ | $D_1$ |
| 2 | $A_2$ | $B_2$ | $C_2$ | $D_2$ |
| 3 | $A_3$ | $B_3$ | $C_3$ | $D_3$ |
| 4 | $A_4$ | $B_4$ | $C_4$ | $D_4$ |
| 5 | $A_5$ | $B_5$ | $C_5$ | $D_5$ |

Figure 18. Simple Two-Dimensional Table

Lists and tables of data can be stored within a data processing system for program reference also, permitting the programmer to instruct the program to perform "table look-up" operations. Such tables are stored in series within the system instead of in the grid-like manner illustrated above. The same table in the data processor might appear as a list, shown in Figure 19.

Even though the table data is stored as a long list, the programmer can still readily specify the required table data in essentially the same manner as a clerk would in instructing another clerk how to use the table first shown. The clerk would specify the table name, then the horizontal row and vertical column headings: TABLE 1, row 3, column C. The GECOM programmer does the same thing in a similar shorthand:

```
        TABLE~1 (3, 3)
meaning TABLE~1, row 3, column 3.
```

Lists, tables, and matrices can all be represented in GECOM source programs and are referred to generically as arrays. A list is a one-dimensional array; a table, two-dimensional.

A three-dimensional array can be depicted graphically as a series of two-dimensional planes; as shown in Figure 20. Three-dimensional arrays could also be represented in storage as a series of sequential lists (one for each plane) like that described for the example above.

Arrays are assigned identifying names by the programmer. To identify array values, subscripts are used to specify rows, columns, and planes.

One-dimensional list = A(I)
Two-dimensional table = A(I,J)
Three-dimensional table = A(I,J,K)

Subscripts can be written as arithmetic expressions, if need be, containing other subscripted arrays, and nested to up to ten deep in any one procedure statement.

```
LIST (A+C)
RATE (A-B*C, L(I,J),X)
```

In the second example A-B*C is the i-subscript, L(I,J) is the j-subscript, and X is the k-subscript for a matrix called RATE. Parentheses are always used to enclose subscripts which must immediately follow the array name.

1 $A_1$ $B_1$ $C_1$ $D_1$ 2 $A_2$ $B_2$ $C_2$ $D_2$ 3 $A_3$ $B_3$ $C_3$ $D_3$ 4 $A_4$ $B_4$ $C_4$ $D_4$ 5 $A_5$ $B_5$ $C_5$ $D_5$

Figure 19. A Two-Dimensional Table in Storage

3110 S3110. ALTER. . . .

This statement sets SW3150 to proceed to S3155 the next time it is processed. SW3150 handles the group suppression of printing of DEPT~NO. When a new department is detected at 3080, it is necessary to print that department number from working storage, but immediately after, blanks are moved to that working storage field (part of the Detail Record) and the MOVE of blanks must be bypassed until the next new department is encountered.

3115 S3115. MOVE. . . .

This statement places the contents of the memory location assigned to hold the job ticket department number to the memory locations assigned to hold the last department number and the working storage department number. The LAST~DEPT is for comparison with the department of the current Job Ticket to determine a change of department at 3080, while the department of working storage is to provide the department number for the first printing of a detail record for a new department, and blanks afterward.

3120 MAN~COUNT=. . . .

This is an assignment statement that sets to zero the memory locations reserved for the named field.

3125 S3125. ADD. . . .

The man count memory location is increased by one.

3130 ADD. . . .

The two named fields are added and the result replaces the previous value of ACC~REG~HRS.

3135 ADD. . . .

The two named fields are added and the result replaces the previous value of ACC~OT~HRS.

3140 IF. . . .

The contents of the LINE~COUNT memory location are compared with the constant, 51. If they are equal, control transfers to procedure statement S3170; if they are not equal, the next statement in sequence is taken (3145). LINE~COUNT = 51 indicates that the last line of a printer page has been printed and a new page (and new headings) must be started.

3145 S3145. WRITE. . . .

The DETAIL RECORD, defined in Data Division statements 4150 through 4180, which includes DEPT, MAN~NBR, NAME, JOB~CODE, REG~HRS, and OT~HRS fields, is printed as a line by the high-speed printer.

3150 SW3150. GO TO. . . .

This is another program switch similar to SW3085 and SW3107. It governs whether the detail record print line contains an actual department number or blanks.

3155 S3155. MOVE. . . .

This statement replaces the contents of the working storage DEPT field with blanks.

3160 ALTER. . . .

This statement changes the object of the GO statement at SW3150 from S3155 to S3075 to bypass S3155 and 3160 until a new department is read.

3165 GO TO. . . .

This statement unconditionally transfers control to S3145.

3170 S3170. PERFORM. . . .

Like statement 3065, this sentence transfers control to the WPH SECTION beginning at 3005. Upon completion of this section, control automatically reverts to the next statement in sequence, 3175. This is used to head up a new page after the capacity of the preceding page has been filled by a department's records.

3175 GO TO. . . .

This statement unconditionally transfers control to S3145.

3180 S3180. ALTER. . . .

This statement changes the object of the GO statement at SW 3107 from S3110 to S3182, so that CLOSE will occur after the final summary card is punched.

3181 GO TO. . . .

This statement unconditionally transfers control to S3100 to compute the final summary card TOTAL ~HRS.

3182 S3182. CLOSE. . . .

This statement terminates processings of the JOB~FILE and the SUMMARY~FILE. The card counts for the card reader and the card punch are printed out on the console typewriter.

In preparing the source program, the programmer may have difficulty in keeping track of codes that of themselves have no meaning. To provide a reference term, he can assign names to them, thusly:

HOURLY = 0
WEEKLY = 1
MONTHLY = 2

Once names are assigned, they can be used in procedure statements within the source program. Such names as those described above are called conditional names for convenience. In actuality, they are special data names, and are formed subject to the same limitations.

CONSTANTS

Data names are generally assigned by the systems programmer to kinds of data, rather than to specific values, because the actual value of the data named is generally a variable (from record to record, for example) or possibly an unknown to be computed by the object program.

Occasionally (even frequently), the programmer will need to place various kinds of specific data in the program - data which remain the same throughout the program. Such constants are designated as literal constants, numeric constants, and figurative constants.

Literal constants are those the programmer intends to use in the program exactly as written. They may be any combination of up to 30 (or 83, depending upon where used) letters, numbers, and symbols of the GECOM character set. To distinguish them from other names, they must be enclosed in quotation marks:

MOVE "FILE~NAME" TO COLUMN~HD.

Literals can be used in output fields to generate headings. They cannot be used in arithmetic calculations.

Numeric constants are comprised of the numerals 0 through 9, plus or minus sign, the letter E for floating-point, and a decimal point. They can be used in three forms of arithmetic calculations: fixed-point, integer, and floating-point.

Fixed-point numerics can contain up to 11 digits, excluding plus or minus sign, and a decimal. Typical fixed-point numerics are:

| | |
|---|---|
| +2.308 | -853.001 |
| 0.03 | 9.11 |

Integers must not exceed 5 digits:

| | |
|---|---|
| 2308 | 85300 |
| 3 | 911 |

For floating-point computations, numerics can be written with mantissas of up to nine digits (one of which must be the left of the decimal) and an exponent between +75 and -75. The largest and smallest floating-point numbers that can be represented are, respectively:

9.99999999E+75 and 0.00000001E-75

If any numeric constant is enclosed in quotation marks, it loses its numeric value and becomes a literal constant.

The constants, 0 through 9 and space (or blank) have been defined within the General Compiler and assigned names. This permits the programmer to use the names within his source program without defining them. These pre-named constants are called figurative constants and are:

| | |
|---|---|
| 0 | ZERO or ZEROES |
| | SPACES |
| 1 | ONE(S) |
| 2 | TWO(S) |
| 3 | THREE(S) |
| 4 | FOUR(S) |
| 5 | FIVE(S) |
| 6 | SIX(ES) |
| 7 | SEVEN(S) |
| 8 | EIGHT(S) |
| 9 | NINE(S) |

Figurative constants may be used in the singular to denote the constant itself or in the plural to imply a string of constants.

EXPRESSIONS

The programmer combines words and symbols into procedure statements to direct computer operations. To facilitate the formulation of such statements showing the relationships and combinations of data names, conditional names, and constants, he has the assistance of arithmetic, relational, and logical expressions.

An arithmetic expression is a sequence of data names, numeric constants, and/or mathematical functions that are combined with symbols which represent arithmetic operations.

Operations and functions available to the programmer and their proper GECOM form are shown in Figure 15. They are listed in priority order, from highest to lowest. All of the listed functions are readily available as part of the GE-225 standard subroutine library and need not be generated during source program compilation or manually by the programmer. Previously-prepared subroutines materially reduce compilation time and programmer effort.

The natural priority of the table can be overridden by parentheses. Parentheses cause the evaluation to be performed from within the innermost set of

3185 STOP RUN "JTS"

This statement is used to generate object program coding for halting processing. In the form used here, the results will be

1. Program halts

2. END is printed by the console typewriter.

3. The literal "JTS" is printed by the console typewriter.

IDENTIFICATION DIVISION PREPARATION

This division enables the programmer to label the source program and provide program identification in the output Edited List.

The Identification Division is prepared on the General Compiler Sentence Form, as illustrated in Figure 35.

Entries for the Job Ticket Summary problem are explained:

1000 IDENTIFICATION DIVISION.

This mandatory heading indicates that entries following are for program identification only. The name should begin in column 8 and be followed by a period.

1005 PROGRAM~ID. JTS.

This entry is mandatory; the name, PROGRAM~ID, should appear beginning in column 8 and followed by a period. The actual program name, JTS, can consist of up to nine typewriter characters followed by a blank, a comma, or a period and can be indented any number of spaces. This name will appear as part of the heading of each page of the Edited List.

1010 AUTHOR. GE CODER

This entry is optional. If used, the sentence name should start in column 8 and be followed by a period. The sentence can be indented as desired, contain up to 30 BCD characters, and ended with a period. If provided, the author's name appears on each page of the Edited List.

1015 DATE COMPILED. JUL. 17

This entry is optional. It can contain up to 30 characters followed by a period. If provided, the compilation date appears on each page of the Edited List.

1020 INSTALLATION. . . .
1025 REMARKS. . . . .

These two sentences, as well as a NEXT~PROGRAM and a SECURITY sentence, are optional. If used, they can contain any information that the programmer wants to appear in the Edited List.

The Identification Division has no effect upon the compilation of the object program, other than that of appearing in the Edited List as described.

## PRODUCING THE OBJECT PROGRAM

Upon completion of the GECOM forms for the source program, the data forms are transcribed to standard punched cards to form the source program deck and organized as shown in Figure 36.

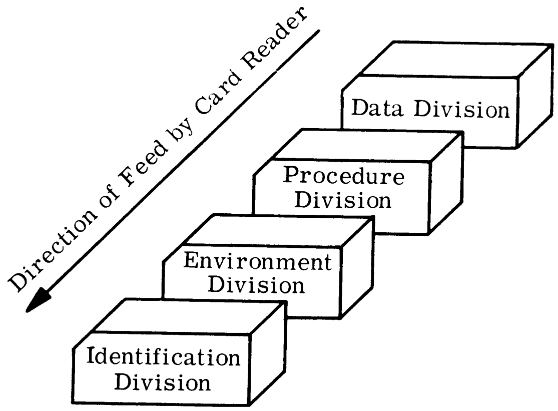


Figure 36. Source Program Deck Organization

A special GECOM call deck is placed before the source program deck and the cards are ready for input to the GE-225 via the card reader.

The minimum GE-225 system configuration for compiling the source program is:

GE-225 Central Processor (with 8192 words of core storage)
Console Typewriter
Card Reader
Card Punch
High-Speed Printer
Magnetic Tape Controller
Four Magnetic Tape Handlers
Five Magnetic Tape Handlers (optional)
Six Magnetic Tape Handlers (optional)

The GECOM Master Tape is mounted on the first magnetic tape handler on the system and includes a library of subroutines that might be required to complete the compiled object program. The source

| VERB | EXAMPLE |
|---|---|
| ADD | ADD TOTL~RECVD TO ON~HAND~QTY |
| ADVANCE | ADVANCE PAY~REGISTER 20 LINES<br>(to slew or skip printer paper) |
| ALTER | ALTER SENT~25 TO PROCEED TO SENT~33.<br>(to change a previously established sequence of operations.) |
| =(Assignment) | QTY~ON~HAND = OLD~QTY + NO~RECVD<br>(to assign an evaluated arithmetic expression to a specified field) |
| CLOSE | CLOSE PAYROL~FILE<br>(to terminate processing of a file) |
| DIVIDE | DIVIDE NUMBER INTO TOTAL GIVING AVERAGE |
| ENTER | ENTER GAP AT ROUTINE~3<br>(to permit insertion of General Assembly Program coding in a GECOM source program.) |
| EXCHANGE | EXCHANGE OLD~TAX, NEW~TAX<br>(to transpose the contents of two fields) |
| GO | GO TO SENT~10<br>(to depart from the normal sequence of operations) |
| IF | IF LINE~COUNT EQ 58 GO TO ADVANCE~PAGE.<br>(to test a condition and transfer to another operation if condition is satisfied) |
| MOVE | MOVE TOTAL TO SAVE~AREA<br>(to transfer data to another location) |
| MULTIPLY | MULTIPLY 0. 18 BY PAY GIVING TAX |
| NOTE | NOTE THIS SENTENCE IS USED FOR CLARITY.<br>(to permit insertion of explanatory text not intended for compilation) |
| OPEN | OPEN ALL INPUT FILES<br>(to initiate file processing) |
| PERFORM | PERFORM FICA~COMP SECTION<br>(to cause execution of a routine in the desired sequence and then return to the sentence following the PERFORM statement.) |
| READ | READ TIME~CARD RECORD<br>(to make input file records available to the program) |
| STOP | STOP<br>(to halt processing of the object program permanently or temporarily.) |
| SUBTRACT | SUBTRACT RECEIPTS OF TRANSAC~FILE FROM ON~ORDER~QTY OF ORDER~FILE GIVING ADJ~ORDER~QTY, IF SIZE ERROR GO TO ZERO~RTN. |
| VARY | VARY CHK~AMT FROM 1 BY 1 UNTIL CHK~AMT GR 5<br>(to initiate and control the repeated execution of the sentence it precedes.) |
| WRITE | WRITE RECORD~1 OF FILE~6<br>(to permit output of data) |

**Figure 14. GECOM Verbs**

The Object Listing includes an ''Input/Output Coding'' print-out showing all input/output file tables, control coding, and service routines. A complete listing of this subsection for the sample problem requires 439 line entries. Part of the Input/Output Coding list is shown in Figure 46.

The final print-out of the Object Listing and the Edited List is ''Location Assignments for GECOM Common Constants,'' Figure 46. This print-out contains the memory locations for object program constants and the compiler-assigned symbols for the constants. For the sample problem, the complete constant listing contains 138 entries.

required subroutines which the operator has previously extracted from the library of subroutines provided. At the user's option, required subroutines can be appended to the object program automatically or manually during compilations.

## GECOM LANGUAGE ELEMENTS

Because the GECOM system was developed with COBOL in mind as the basic programming language, the GECOM language elements most closely resemble those of the COBOL language. Also, because the intent is to provide English-language programming, GECOM elements parallel those of English.

GECOM has a basic vocabulary consisting of words and symbols; it has rules of grammar or syntax; and it has punctuation symbols for clarity. In each case, there is greater simplicity than in English: the vocabulary is small; the rules of grammar are simple, yet precise; the use of punctuation is limited. These are true because the demands placed upon the user are kept simple and unambiguous. The source programming language is required to state facts and give instructions clearly and specifically; it is a language of command, not narration, and thus consists primarily of verbs and nouns. These can be formed into simple and complex sentences usually intelligible without special training, although sentences acceptable to the General Compiler cannot be written without familiarity with the grammar.

Words and symbols are the tools of the GECOM programmer and are composed of individual letters, numbers, and special characters. The basic character set of GECOM and equivalent GE-225 character codes are illustrated in the accompanying table, Figure 13. Special character sets are available for the printer.

Many of the basic characters, in addition to being used in words, have special meanings for GECOM; these will be discussed where appropriate.

Words, in GECOM, are divided into two major groups - names and verbs.

### VERBS

As in English, verbs denote action; unlike English, GECOM verbs are never taken in the passive voice, the narrative or declarative sense, or in any tense other than the present tense. Each verb that the programmer uses in the source program (except the verb NOTE) will have some effect in the object program.

Most verbs will be reflected directly in the machine-language coding of the compiled object program; others do not appear in the object program, but do act with the compiler to construct the object program.

Certain words that, in English, are not verbs are considered as such by the General Compiler. The most commonly-used and most useful of these is the word, IF, which is used in expressing conditions, relationships, and comparisons. For example, in the expressions:

IF NOT END OF FILE, GO TO . . . . . . . . . .
OR
IF A EQUALS B, GO TO . . . . . . . . . . . .

IF causes a comparison between the actual condition and the stated END OF FILE condition or, in the second example, causes a comparison between A and B. Such near-verbs will be discussed as if they were verbs.

The GECOM verbs and examples of how each might be used are listed in Figure 14.

### NAMES

Most words in the GECOM source program will be names. The programmer is preparing a program for handling data, but is not concerned with the actual data itself; he is more concerned with preparing data manipulation procedures, but once they are written they are only of as much importance as the data they manipulate. For these reasons, and to take advantage of the leverage that GECOM provides, the programmer will refer to data and previously written procedures by name whenever possible.

Names can be readily grouped by type and fall within these groups:

1. Data Names
2. Procedure Names
3. Conditional Names
4. Constants

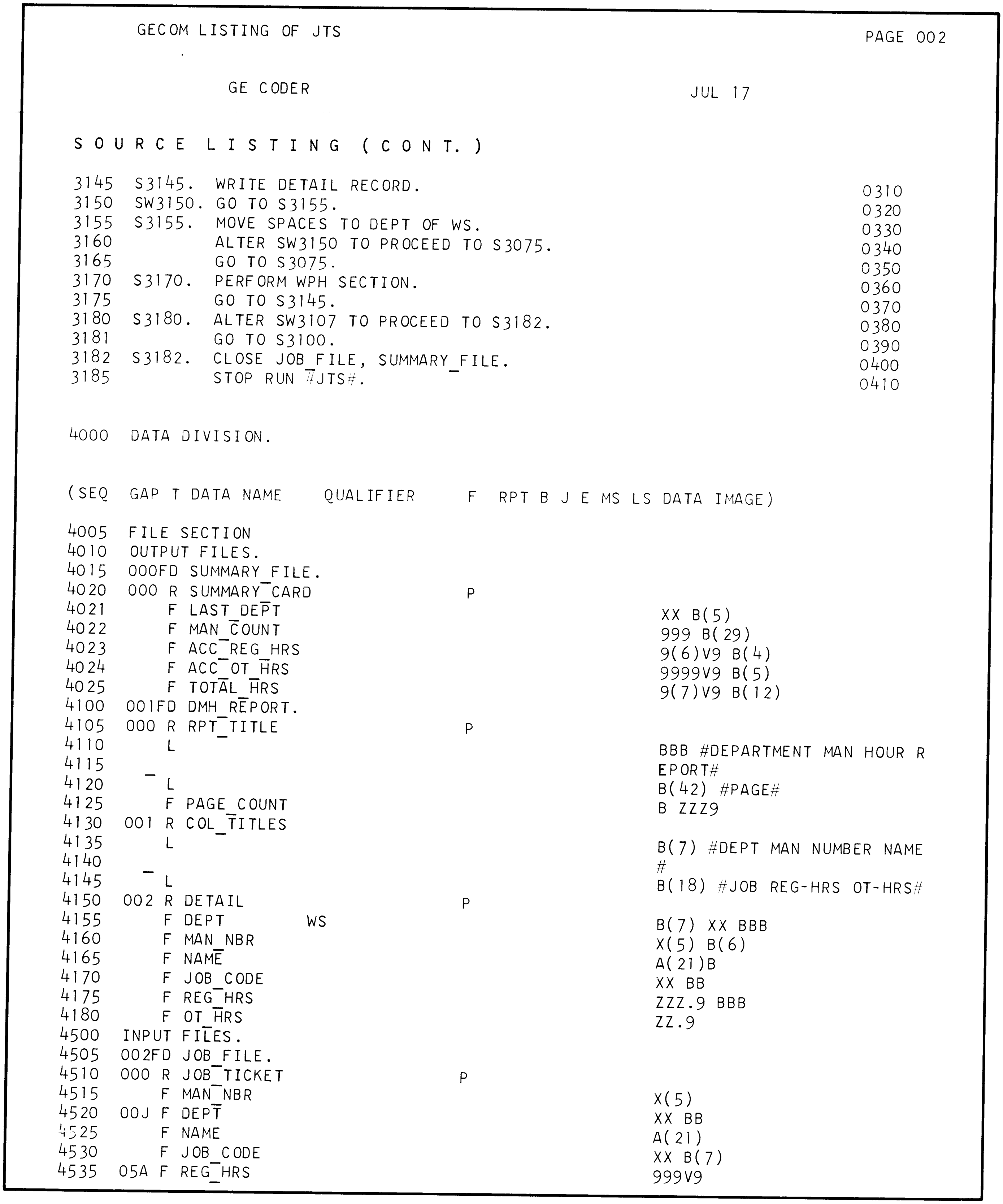
### DATA NAMES

Data names represent data to be used in an object program, and are programmer-assigned, not to specific data, but to kinds of data. For example, in a file processing application, data names would be assigned to all input and output files, such as:

MASTER~FILE
TRANSACTIONS
PRINT~FILE
etc.

and, within a file, records would bear data names, such as:

STOCK~RCD
PAY~RCD
INV~RCD~1
etc.



```
       GECOM LISTING OF JTS                                         PAGE 002

                 GE CODER                                    JUL 17

   S O U R C E   L I S T I N G   ( C O N T. )

   3145  S3145.   WRITE DETAIL RECORD.                                 0310
   3150  SW3150.  GO TO S3155.                                         0320
   3155  S3155.   MOVE SPACES TO DEPT OF WS.                           0330
   3160           ALTER SW3150 TO PROCEED TO S3075.                    0340
   3165           GO TO S3075.                                         0350
   3170  S3170.   PERFORM WPH SECTION.                                 0360
   3175           GO TO S3145.                                         0370
   3180  S3180.   ALTER SW3107 TO PROCEED TO S3182.                    0380
   3181           GO TO S3100.                                         0390
   3182  S3182.   CLOSE JOB_FILE, SUMMARY_FILE.                        0400
   3185           STOP RUN #JTS#.                                      0410


   4000  DATA DIVISION.


   (SEQ  GAP T DATA NAME    QUALIFIER     F  RPT B J E MS LS DATA IMAGE)

   4005  FILE SECTION
   4010  OUTPUT FILES.
   4015  000FD SUMMARY_FILE.
   4020  000 R SUMMARY_CARD                P
   4021      F LAST_DEPT                                       XX B(5)
   4022      F MAN_COUNT                                       999 B(29)
   4023      F ACC_REG_HRS                                     9(6)V9 B(4)
   4024      F ACC_OT_HRS                                      9999V9 B(5)
   4025      F TOTAL_HRS                                       9(7)V9 B(12)
   4100  001FD DMH_REPORT.
   4105  000 R RPT_TITLE                   P
   4110      L                                                 BBB #DEPARTMENT MAN HOUR R
   4115                                                        EPORT#
   4120    - L                                                 B(42) #PAGE#
   4125      F PAGE_COUNT                                      B ZZZ9
   4130  001 R COL_TITLES
   4135      L                                                 B(7) #DEPT MAN NUMBER NAME
   4140                                                        #
   4145    - L                                                 B(18) #JOB REG-HRS OT-HRS#
   4150  002 R DETAIL                      P
   4155      F DEPT          WS                                B(7) XX BBB
   4160      F MAN_NBR                                         X(5) B(6)
   4165      F NAME                                            A(21)B
   4170      F JOB_CODE                                        XX BB
   4175      F REG_HRS                                         ZZZ.9 BBB
   4180      F OT_HRS                                          ZZ.9
   4500  INPUT FILES.
   4505  002FD JOB_FILE.
   4510  000 R JOB_TICKET                  P
   4515      F MAN_NBR                                         X(5)
   4520  00J F DEPT                                            XX BB
   4525      F NAME                                            A(21)
   4530      F JOB_CODE                                        XX B(7)
   4535  05A F REG_HRS                                         999V9
```

Figure 38. Edited List

The Data Division Form, Figure 8, is used exclusively for describing data to be used in the object program. Headings are provided to guide the proper placement of data. These are discussed in the later section, Data Division Preparation.

The Sentence Form, Figure 9, is used for the preparation of data for the Identification, Environment, and Procedure Divisions. Headings, which would add little, are omitted. Rules for Sentence Form preparation are few and simple.

Where applicable, such rules are discussed in the section, ''Application of Basic GECOM,'' along with the preparation of the four divisions of the source program. The fourth major tool provided by the GECOM system, is the General Compiler itself. Examination shows considerable similarity between the General Compiler program and a complex business data processing object program.

1. The General Compiler operates upon input: the source-language program.

2. Compiler processing consists of repetitive runs of a set of instructions: the General Compiler.

3. It produces an output: the object program.

4. It produces reports: the Edited List and error messages.

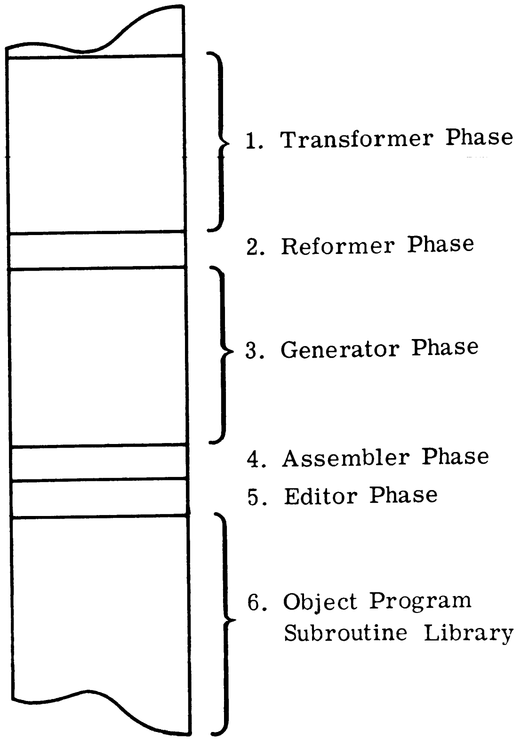


Figure 11. General Compiler Program Organization

Figure 10 illustrates, in broad terms, the relationships between the programmer-produced source programs, the General Compiler, the computer, and the output object program.

Up to this point, the General Compiler has been discussed as if it were a single program, and it can still be considered as such. Conversely, it can also be considered to be a series of sequential programs as illustrated in Figure 11. Note that there are five major groupings: Transformer, Reformer, Assembler, Editor, and Subroutines.

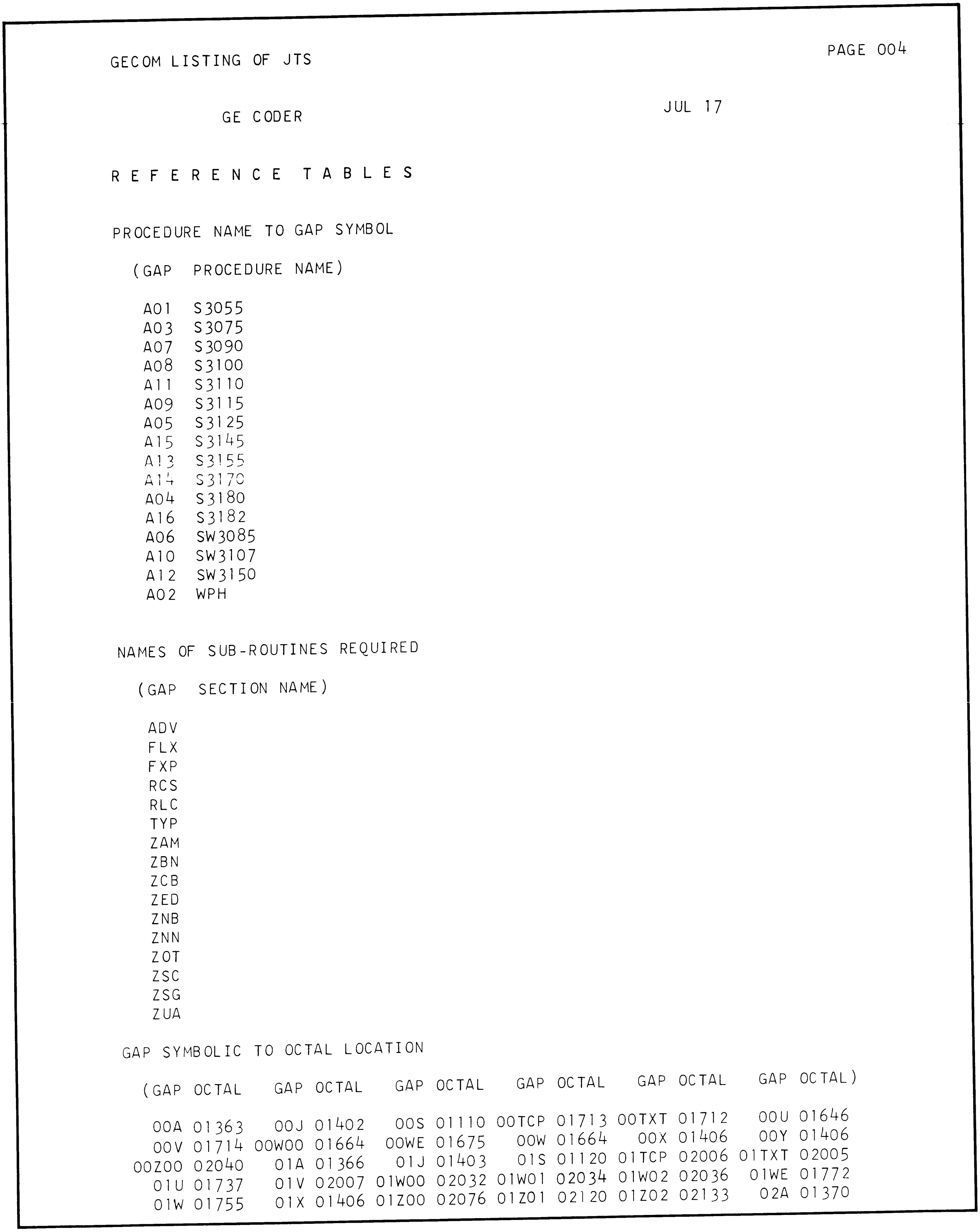
The transformer phase translates the source program into an intermediate internal language suitable for processing, prints out Identification and Environment Divisions as required, groups and organizes Procedure and Data Division material for further processing while checking for validity and consistency, prints error messages, screens out unessential optional words, and initiates the preparation of the object program.

The reformer phase is essentially executive in that it calls forth from the generator library (also a part of the Compiler) those routines required to produce the object program.

The assembler phase translates from the intermediate language, assembles the coding into machine language, and produces the completed object program either in punched cards or on magnetic tape.

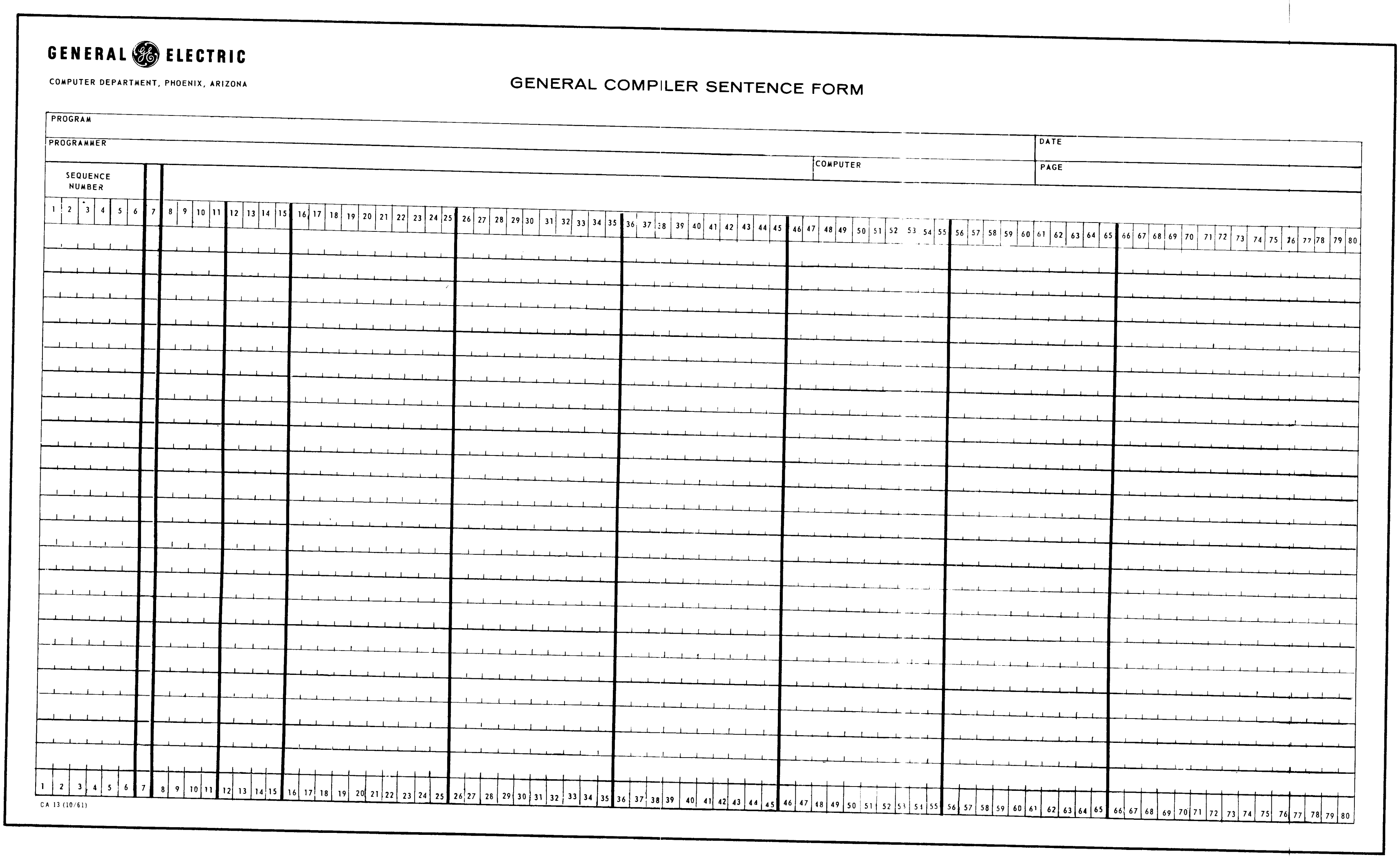
The editor phase provides the documentation of the program in the form of the Edited List. This includes a print-out of the entire original source program, a merged list showing the generated symbolic coding and the machine-language coding, and cross-reference tables. Additionally, it lists, from the master list of subroutines below, those required to complete the object program. Examples of the Edited List are included in the section, ''Application of Basic GECOM.''

The subroutine library is a collection of previously-prepared subroutines common to most object programs that may be required to complete the object program. While these could be produced during compilations, to reduce compilation time and avoid repetitive processing during compiling, the General Compiler shows (on the Edited List) all such subroutines which will be needed when the object program is run. A special program loading routine will place into memory the object program and the



```
GECOM LISTING OF JTS                                        PAGE 004

          GE CODER                                 JUL 17

R E F E R E N C E   T A B L E S

PROCEDURE NAME TO GAP SYMBOL

  (GAP  PROCEDURE NAME)

   A01  S3055
   A03  S3075
   A07  S3090
   A08  S3100
   A11  S3110
   A09  S3115
   A05  S3125
   A15  S3145
   A13  S3155
   A14  S3170
   A04  S3180
   A16  S3182
   A06  SW3085
   A10  SW3107
   A12  SW3150
   A02  WPH

NAMES OF SUB-ROUTINES REQUIRED

  (GAP  SECTION NAME)

   ADV
   FLX
   FXP
   RCS
   RLC
   TYP
   ZAM
   ZBN
   ZCB
   ZED
   ZNB
   ZNN
   ZOT
   ZSC
   ZSG
   ZUA

GAP SYMBOLIC TO OCTAL LOCATION

  (GAP OCTAL    GAP OCTAL    GAP OCTAL    GAP OCTAL    GAP OCTAL    GAP OCTAL)

   00A 01363    00J 01402    00S 01110 00TCP 01713 00TXT 01712    00U 01646
   00V 01714 00W00 01664   00WE 01675    00W 01664    00X 01406    00Y 01406
 00Z00 02040    01A 01366    01J 01403    01S 01120 01TCP 02006 01TXT 02005
   01U 01737    01V 02007 01W00 02032 01W01 02034 01W02 02036   01WE 01772
   01W 01755    01X 01406 01Z00 02076 01Z01 02120 01Z02 02133    02A 01370
```

Figure 40. Edited List



GENERAL ELECTRIC

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL COMPILER SENTENCE FORM

PROGRAM

PROGRAMMER

DATE

COMPUTER

PAGE

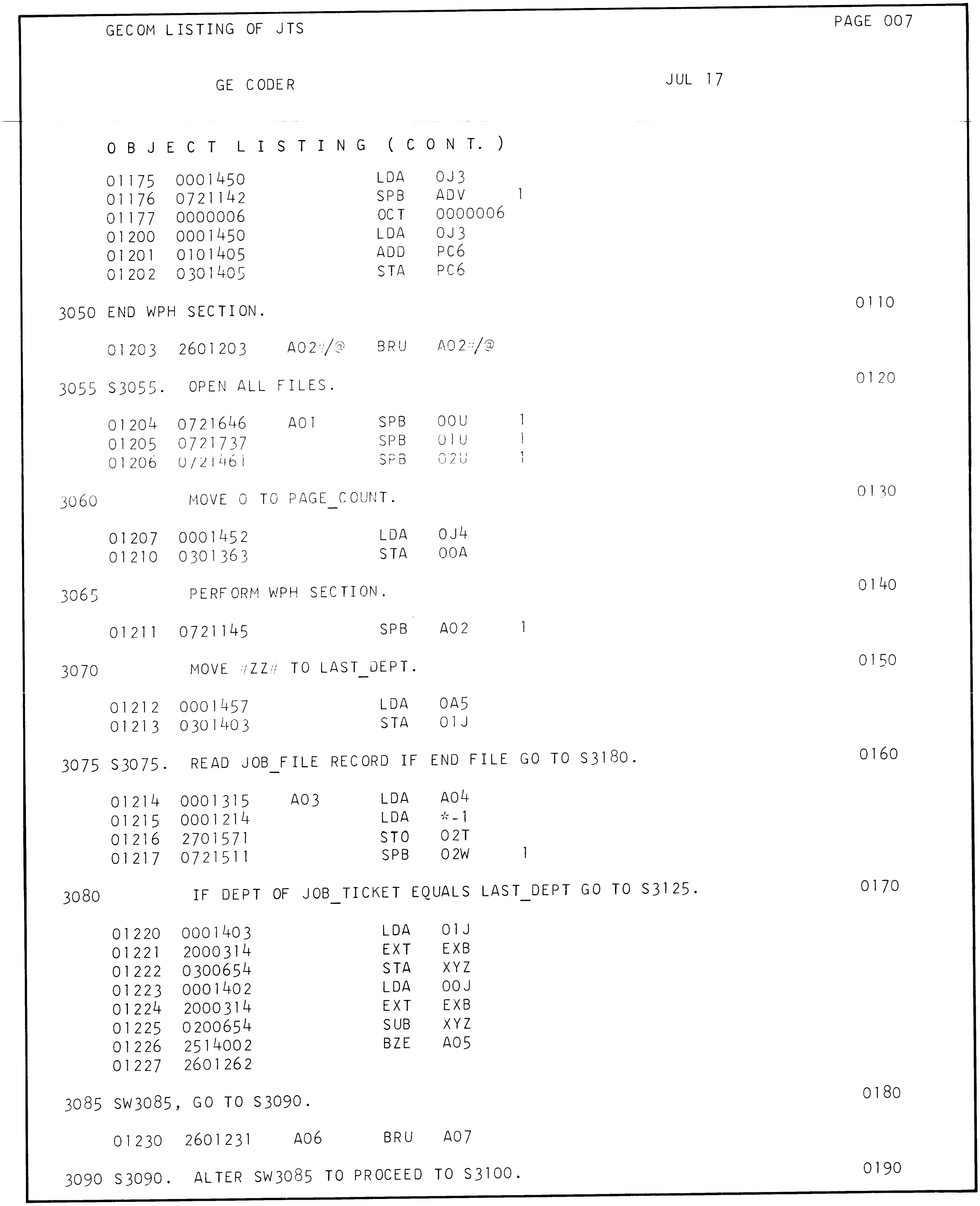
SEQUENCE NUMBER

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 51 51 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

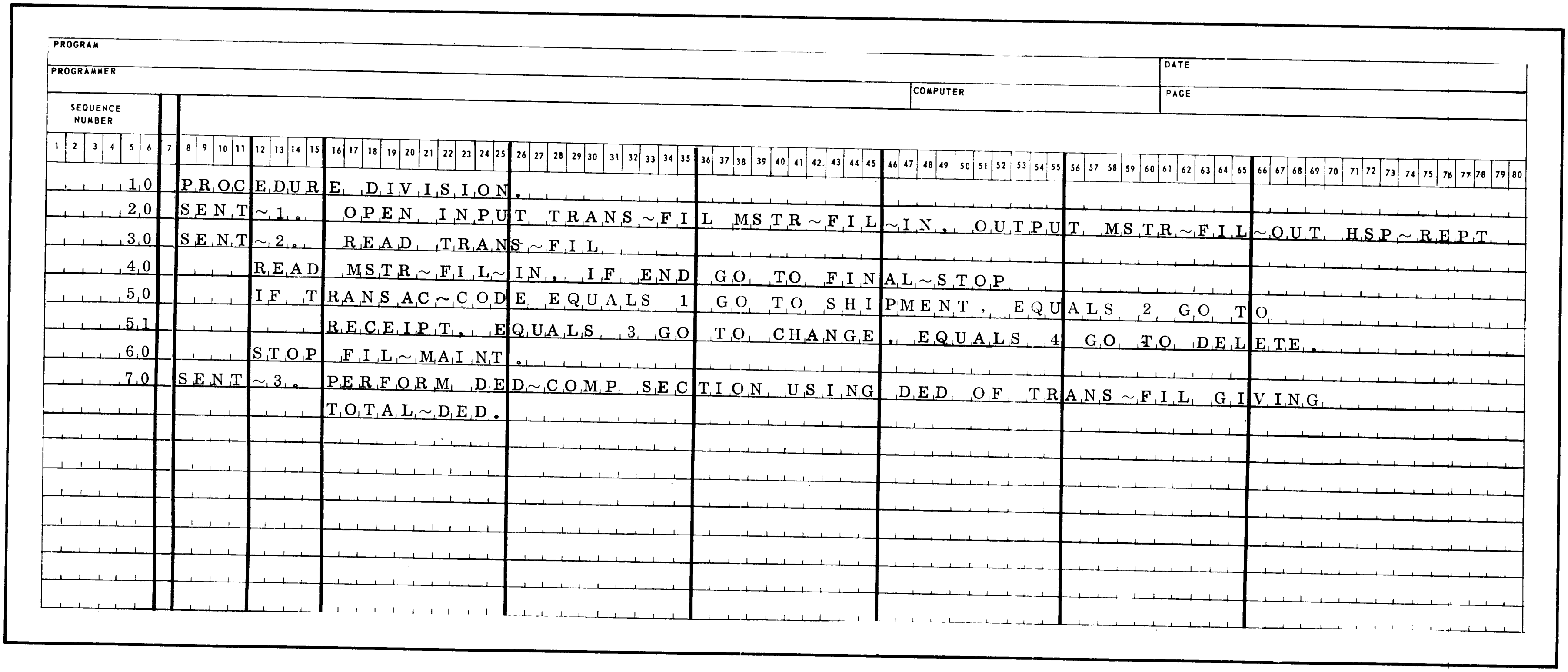
CA 13 (10/61)

Figure 9. The GECOM Sentence Form



```
         GECOM LISTING OF JTS                                                PAGE 007

                      GE CODER                                   JUL 17

         O B J E C T   L I S T I N G   ( C O N T. )

         01175  0001450                LDA    0J3
         01176  0721142                SPB    ADV       1
         01177  0000006                OCT    0000006
         01200  0001450                LDA    0J3
         01201  0101405                ADD    PC6
         01202  0301405                STA    PC6

    3050 END WPH SECTION.                                                   0110

         01203  2601203     A02#/@     BRU    A02#/@

    3055 S3055.  OPEN ALL FILES.                                            0120

         01204  0721646     A01        SPB    00U       1
         01205  0721737                SPB    01U       1
         01206  0721461                SPB    02U       1

    3060         MOVE 0 TO PAGE_COUNT.                                      0130

         01207  0001452                LDA    0J4
         01210  0301363                STA    00A

    3065         PERFORM WPH SECTION.                                       0140

         01211  0721145                SPB    A02       1

    3070         MOVE #ZZ# TO LAST_DEPT.                                    0150

         01212  0001457                LDA    0A5
         01213  0301403                STA    01J

    3075 S3075.  READ JOB_FILE RECORD IF END FILE GO TO S3180.              0160

         01214  0001315     A03        LDA    A04
         01215  0001214                LDA    *-1
         01216  2701571                STO    02T
         01217  0721511                SPB    02W       1

    3080         IF DEPT OF JOB_TICKET EQUALS LAST_DEPT GO TO S3125.        0170

         01220  0001403                LDA    01J
         01221  2000314                EXT    EXB
         01222  0300654                STA    XYZ
         01223  0001402                LDA    00J
         01224  2000314                EXT    EXB
         01225  0200654                SUB    XYZ
         01226  2514002                BZE    A05
         01227  2601262

    3085 SW3085, GO TO S3090.                                               0180

         01230  2601231     A06        BRU    A07

    3090 S3090.  ALTER SW3085 TO PROCEED TO S3100.                          0190
```

**Figure 42. Edited List**



PROGRAM

PROGRAMMER

DATE

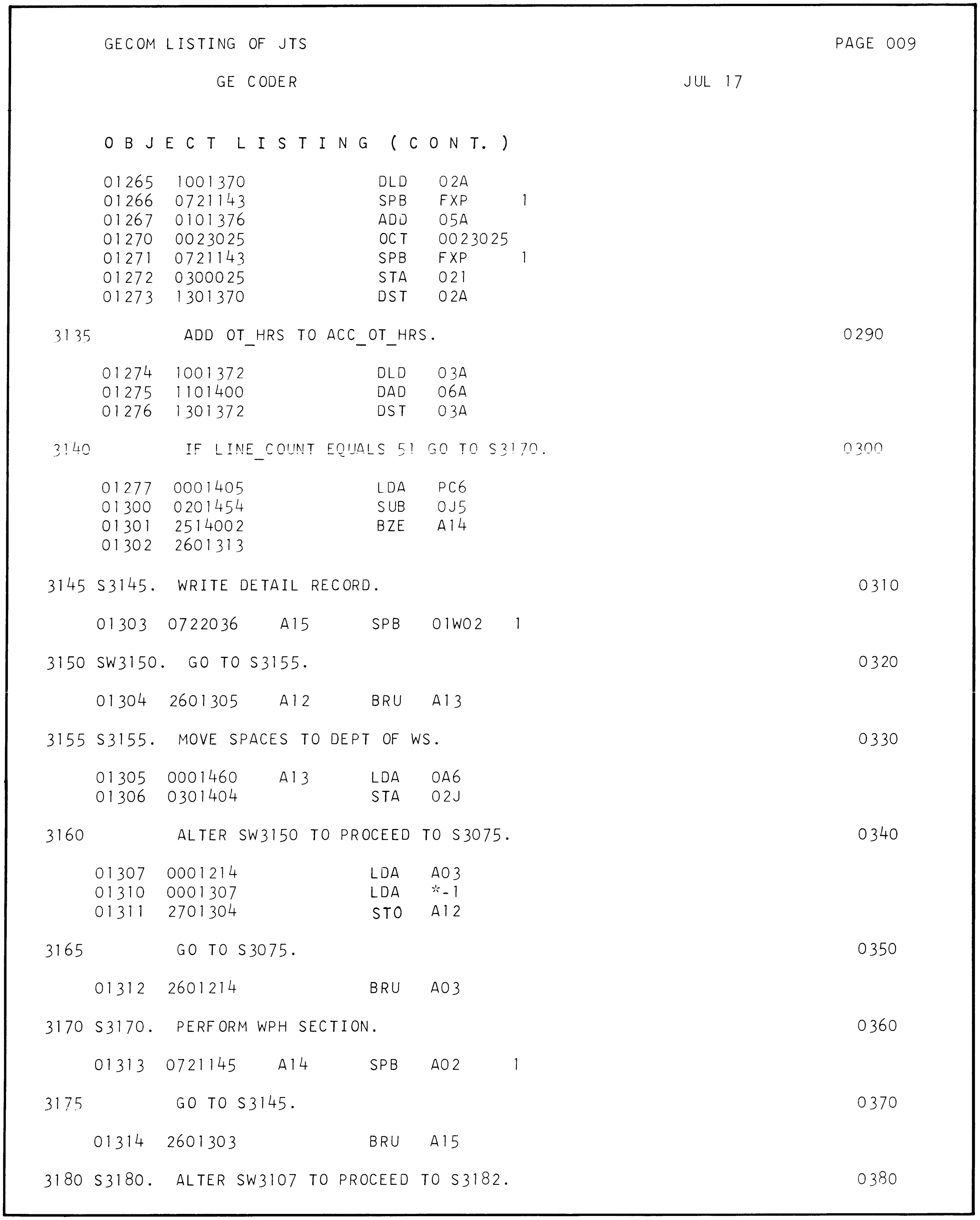
COMPUTER

PAGE

SEQUENCE NUMBER

```
   1.0 PROCEDURE DIVISION.
   2.0 SENT~1.  OPEN INPUT TRANS~FIL MSTR~FIL~IN, OUTPUT MSTR~FIL~OUT HSP~REPT
   3.0 SENT~2.  READ TRANS~FIL
   4.0     READ MSTR~FIL~IN, IF END GO TO FINAL~STOP
   5.0     IF TRANSAC~CODE EQUALS 1 GO TO SHIPMENT, EQUALS 2 GO TO
   5.1         RECEIPT, EQUALS 3 GO TO CHANGE, EQUALS 4 GO TO DELETE.
   6.0     STOP FIL~MAINT.
   7.0 SENT~3. PERFORM DED~COMP SECTION USING DED OF TRANS~FIL GIVING
               TOTAL~DED.
```

Figure 7. Procedure Division Layout



```
     GECOM LISTING OF JTS                                                       PAGE 009

              GE CODER                                                  JUL 17

     O B J E C T   L I S T I N G   ( C O N T. )

     01265  1001370             DLD   02A
     01266  0721143             SPB   FXP      1
     01267  0101376             ADD   05A
     01270  0023025             OCT   0023025
     01271  0721143             SPB   FXP      1
     01272  0300025             STA   021
     01273  1301370             DST   02A

3135          ADD OT_HRS TO ACC_OT_HRS.                                        0290

     01274  1001372             DLD   03A
     01275  1101400             DAD   06A
     01276  1301372             DST   03A

3140          IF LINE_COUNT EQUALS 51 GO TO S3170.                             0300

     01277  0001405             LDA   PC6
     01300  0201454             SUB   0J5
     01301  2514002             BZE   A14
     01302  2601313

3145 S3145.  WRITE DETAIL RECORD.                                               0310

     01303  0722036   A15       SPB   01W02    1

3150 SW3150.  GO TO S3155.                                                      0320

     01304  2601305   A12       BRU   A13

3155 S3155.  MOVE SPACES TO DEPT OF WS.                                         0330

     01305  0001460   A13       LDA   0A6
     01306  0301404             STA   02J

3160          ALTER SW3150 TO PROCEED TO S3075.                                 0340

     01307  0001214             LDA   A03
     01310  0001307             LDA   *-1
     01311  2701304             STO   A12

3165          GO TO S3075.                                                      0350

     01312  2601214             BRU   A03

3170 S3170.  PERFORM WPH SECTION.                                               0360

     01313  0721145   A14       SPB   A02      1

3175          GO TO S3145.                                                      0370

     01314  2601303             BRU   A15

3180 S3180.  ALTER SW3107 TO PROCEED TO S3182.                                  0380
```

Figure 44. Edited List

5. Elements. In a few cases, for convenience, fields are further subdivided into "elements." For example, a part numbering system could be so organized that portions of the part number had added significance. For example: 18253702, NPN Transistor; 18 meaning electrical, 2 meaning a component (not a subassembly), 53 meaning tubes and solid-state devices, and 702 to identify the particular item.

The relationship between these various data levels are readily shown:

```
FILE
    RECORD
        GROUP 1
        GROUP 2
            FIELD
            FIELD
                ELEMENT
                ELEMENT
            FIELD
        GROUP 3
        GROUP 4
```

As mentioned earlier, all data to be used or created by the object program must be defined. A typical Data Division for GECOM is shown in Figure 6, giving representative examples of data definitions. The Data Division for a representative problem is presented and explained in the section, "Application of Basic GECOM". The relationship between Data Division and input data is also shown in Figure 6.

The Procedure Division, Figure 7, indicates the steps that the programmer wishes the object program to accomplish. These steps are expressed in English words, symbols, and sentences that have meaning to the General Compiler. Although the steps described in the Procedure Division closely parallel those of the eventual object program, it is misleading to consider the Procedure Division alone to be the source program. The source program is not complete without Data, Environment, and Identification Divisions.

Sentences in the Procedure Division invariably contain verbs to denote the desired action, names (of data, constants, etc.) or operands to show what is to be acted upon, and various modifiers for clarity. Sentences can be grouped into sections to facilitate reference and permit the performance of a series of sentences out of the normal sequence.

Procedure statements or sentences can be simple:

ADD 0.5, RATE OF PAY~FILE.

This will create coding in the object program to add the constant 0.5 to whatever value (of the RATE from the PAY~FILE) had been read into the computer. Or statements can be highly complex, involving several clauses and modifiers, such as:

IF PART~NUMBER OF MSTR~INVNTRY IS LESS THAN PART~NUMBER OF TRANSACTIONS GO TO WRITE~MASTER, IF EQUAL GO TO UPDAT~MASTER, IF GREATER GO TO NEW~RECORD.

This statement would result in object program coding to cause the following:

1. The part number of the master inventory record (previously read in) would be compared with the part number of the current transaction record.

2. If the part number of the master inventory record is:

a. the lesser of the two, program control is transferred to a routine called WRITE~MASTER, which causes the master inventory record to be written out as part of a master file,

b. equal to the transaction part number, program control is transferred to a routine called UPDAT~MASTER, which modifies the master inventory record in some manner,

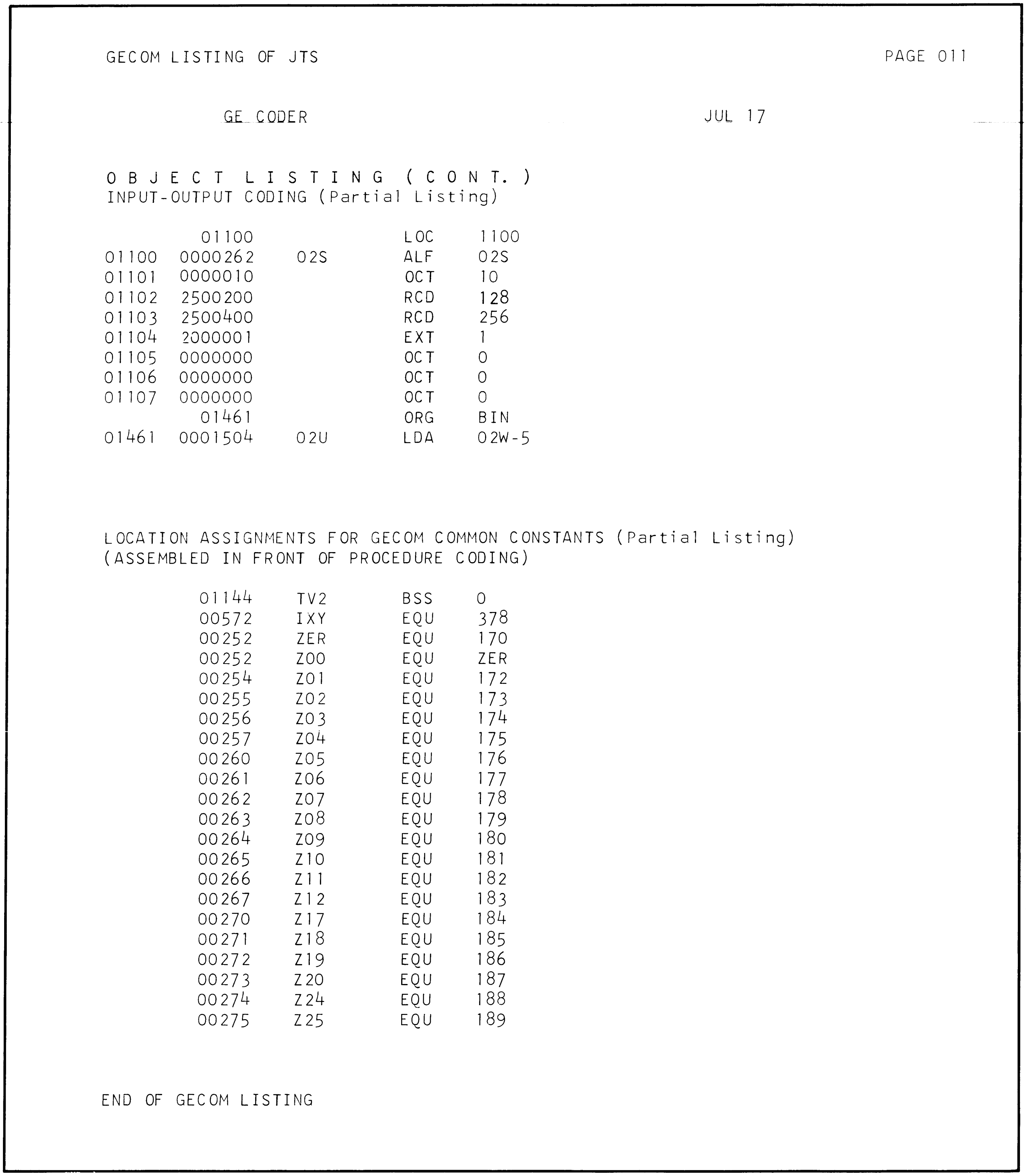
c. the greater of the two, program control transfers to a routine called NEW~RECORD, which causes a new record to be added to the master file.

Procedure Division sentences are performed in the sequence in which they appear, unless that sequence is modified by a "GO" or a "PERFORM" statement as explained in the next section of this chapter, "GECOM Language Elements".

Typical Procedure Division statements are illustrated in Figure 13. Note that sentences can be named (for reference to them by other sentences) or unnamed. Lines 20, 30 and 70 have been named SENT~1, SENT~2, and SENT~3, although more descriptive names can be assigned at the programmer's discretion. More detailed information for preparing a source program Procedure Division is covered in the section, "Application of Basic GECOM".

In addition to LANGUAGE and ORGANIZATION, the third item that the GECOM system provides for the programmer is a set of forms to facilitate source program preparation and documentation. Two basic forms are provided, the General Compiler Data Division Form, number CA-14, and the General Compiler Sentence Form, number CA-13.

Both forms are designed to make it easy to translate the programmer-prepared source program information into a machine-readable form, such as punched cards or paper tape. Each horizontal line of either form provides for up to 80 units of information, corresponding to 80 punched card columns.



```
GECOM LISTING OF JTS                                                PAGE 011

        GE CODER                                         JUL 17

OBJECT LISTING (CONT.)
INPUT-OUTPUT CODING (Partial Listing)

          01100                 LOC    1100
01100   0000262      02S        ALF    02S
01101   0000010                 OCT    10
01102   2500200                 RCD    128
01103   2500400                 RCD    256
01104   2000001                 EXT    1
01105   0000000                 OCT    0
01106   0000000                 OCT    0
01107   0000000                 OCT    0
          01461                 ORG    BIN
01461   0001504      02U        LDA    02W-5



LOCATION ASSIGNMENTS FOR GECOM COMMON CONSTANTS (Partial Listing)
(ASSEMBLED IN FRONT OF PROCEDURE CODING)

        01144        TV2        BSS    0
        00572        IXY        EQU    378
        00252        ZER        EQU    170
        00252        Z00        EQU    ZER
        00254        Z01        EQU    172
        00255        Z02        EQU    173
        00256        Z03        EQU    174
        00257        Z04        EQU    175
        00260        Z05        EQU    176
        00261        Z06        EQU    177
        00262        Z07        EQU    178
        00263        Z08        EQU    179
        00264        Z09        EQU    180
        00265        Z10        EQU    181
        00266        Z11        EQU    182
        00267        Z12        EQU    183
        00270        Z17        EQU    184
        00271        Z18        EQU    185
        00272        Z19        EQU    186
        00273        Z20        EQU    187
        00274        Z24        EQU    188
        00275        Z25        EQU    189


END OF GECOM LISTING
```

Figure 46. Edited List

procedure. In addition, standardization of divisions, sections, procedure statements, and other program elements facilitates communication between programmers and permits program debugging in the same language in which the program was written.

The four divisions of a GECOM source program are:

1. The Identification Division
2. The Environment Division
3. The Data Division
4. The Procedure Division

The Identification Division, Figure 4, provides the programmer with the means for labelling and describing the source program in English-language form. In addition to the program name, author (programmer) and date compiled, this division can include other pertinent information, such as next-program-in-sequence, security classification, location, and explanatory comments as needed. During compilation, this data becomes the label for the object program and is automatically reproduced on output listings, such as the Edited List.

Programmer use of the Identification Division is flexible. The only portion required by the General Compiler is the division name and the PROGRAM ID sentence; all other sentences are at the programmer's option.

Preparation of the Identification Division is discussed further in the section, Application of Basic GECOM.

The Environment Division, Figure 5, provides a link between the source program and the data processing equipment. It defines the computer system configuration and its relationship to the source and object program. The General Compiler depends upon the Environment Division to provide information which associates input and output equipment with the data names for each file to be used in processing. The information in the Environment Division is specified by the systems programmer in English language clauses.

In preparing the Environment Division, the programmer enters the information in a predetermined way. This format is sectionalized under four sentence headings as described below:

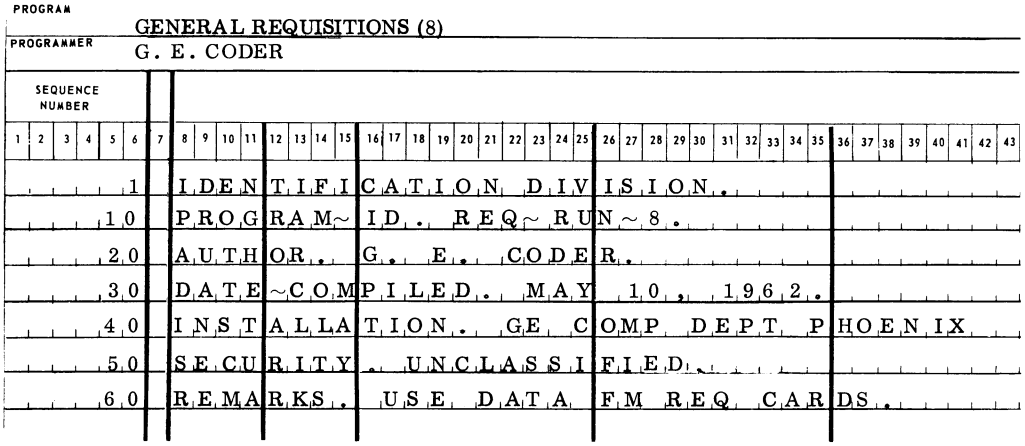
1. The OBJECT~COMPUTER sentence, the first entry, is used to describe the computer on which the object program is to be run.

2. The I~O~CONTROL (input/output control) sentence, the second entry, specifies nonstandard error and tape label checking procedures. In addition, programming control is facilitated by permitting the specification of program rerun points, memory dump assignments, and identification of multifile magnetic tape reels.

3. The third sentence, FILE CONTROL, identifies input/output files and provides for their assignment to specific input/output units.

4. The COMPUTATION~MODE sentence assigns the internal mode of calculation. Sentence use is optional; it is used only when it is desired that computation occur in the floating-point mode, either programmed or in the optional Auxiliary Arithmetic Unit.

The accompanying example illustrates typical entries describing the environment for a representative program. Entry 10 describes the data processing system for which the object program is intended: a GE-225 system with two memory modules (8192 words of core storage), one card reader, one card



```
PROGRAM     GENERAL REQUISITIONS (8)
PROGRAMMER  G. E. CODER

SEQUENCE
NUMBER
1 2 3 4 5 6  7  8 ...
          1     IDENTIFICATION DIVISION.
         10     PROGRAM~ID. REQ~RUN~8.
         20     AUTHOR. G. E. CODER.
         30     DATE~COMPILED. MAY 10, 1962.
         40     INSTALLATION. GE COMP DEPT PHOENIX
         50     SECURITY. UNCLASSIFIED.
         60     REMARKS. USE DATA FM REQ CARDS.
```

Figure 4. Identification Division Layout

# APPENDIX 1. THE GENERAL COMPILER VOCABULARY

Words and terms that appear in the following list must be considered to be part of the General Compiler vocabulary and must not be used by the systems programmer in forming data or procedure names, nor may they be used in any manner in a source program other than as provided by the GECOM Language Specifications.

Where warranted, many of the terms have been defined or explained. Terms not so explained were deemed to be self-evident in meaning. In addition, the body of the manual contains many examples that illustrate the use of most of the vocabulary terms.

ABS - Absolute value, or magnitude, of a number, regardless of sign.

ACCESS - Part of descriptive name Mass Random Access Data Storage.

ADD - To add two quantities and store the sum in either the last-named field or the specified field.

ADVANCE - To vertically skip or slew the printer paper.

AFTER

ALL

ALTER - To modify a sequence of operations specified in one or more GO sentences.

AND - A logical operator.

ARE

ARRAY - A multi-valued field that may be referenced by name and subscript. An array may be one, two, or three dimensional and may have corresponding number of subscripts. An array must be defined in the Array Section of the Data Division.

ASSIGN - To direct the placement of a file or program to an input-output media.

ASSIGNMENT - To evaluate an arithmetic expression and assign the result to a field. To equate data names.

ATAN - Arc tangent. A mathematical function that may be used within arithmetic expressions. Calculated in floating point arithmetic.

AUTHOR - An optional Identification Division sentence name.

BEGIN - Entrance point to a source program section.

BEGINNING

BGN~FIL~LABL - A tape record preceding each file of a multi-file tape.

BGN~TAP~LABL - The first record on any tape except in multi-file tape.

BINARY - Pertaining to the binary number system, as opposed to decimal or binary coded decimal.

BLOCK - See Glossary

BUFFER - A device which stores data temporarily during transfer operations.

BY

CARD

CLOSE - To terminate processing of input or output reels and files with optional rewind and/or lock.

COMMON (~STORAGE) - An optional Data Division Section name.

COMPUTATION ~ MODE - An optional Environment Division sentence name.

CONSTANT - An optional Data Division section name.

CONTAINS

CONTROL - Interpretation and execution of operations.

CONTROL~KEY - The field or fields by which a record is identified.

COPY - To duplicate from another area.

# THE BASIC GECOM SYSTEM

## GENERAL

For clarity and simplicity, only the Basic GECOM system is described in this section. Brief descriptions of extensions to Basic GECOM are provided in the section, "Extension to GECOM". These extensions, for the most part, expand the capabilities of GECOM to encompass recent language developments.

Implementing a data processing application on a computer involves a broad procedure that has been outlined as follows:

1. Define the problem

2. Determine the procedure to be followed in solving the problem

3. Prepare the computer program, including testing

4. Run the program on the computer with appropriate input data.

If the programmer has at his disposal the automatic coding system of GECOM, the above procedure becomes:

1. Define the problem

2. Determine the procedure to be followed in solving the problem

3. Prepare the source program in problem-oriented language

4. Compile the object program from the source program, using the General Compiler

5. Machine-test (debug) the object program

6. Run the object program on the GE-225 with appropriate input data.

At first glance, automatic coding seemingly complicates the task of data processing. However, as shown in Figure 3, the burden on the programmer is no greater, and often is appreciably less. For example, the step from item 2 to item 3, above, is greatly facilitated by the GECOM-provided ability to express procedural steps in English language statements. Additionally, each statement the programmer writes is several times more powerful than the machine-language or symbolic instructions that he would otherwise use. Also, he is materially assisted in the machine-test or check-out phase, item 5, by the assistance provided by the General Compiler in the form of detailed print-outs of error conditions and of the complete compilation process. The print-outs are as easy to read as the programmer-prepared procedure statements of the source program.

This section is devoted primarily to discussion of item 3, source program preparation, using the GECOM system. Incidental references will be made to the other areas, such as the compilation process, as required.

Assuming that a well-defined data processing problem has been assigned to a systems programmer, he determines the detailed procedures for problem solution and generally prepares a flow chart describing those procedures. Flow charts can be broad or detailed, depending upon the problem and the programmer. Invariably, they are sufficiently detailed to serve as a guide for programming the problem solution. The section, "Application of Basic GECOM." illustrates typical flow charts.

## GECOM SYSTEM COMPONENTS

With these preliminaries out of the way, the programmer is ready to prepare the source program. What does the GECOM system provide him to assist in this task?

First, it provides him the necessary language that eliminates tedious machine-language or symbolic coding. Language is discussed in the following section, "GECOM Language Elements".

Second, it provides him with a standard source program organization, which corresponds to the format followed by the compilation output. GECOM source programs are partitioned into four divisions, intended for separate and independent preparation. This facilitates changes; if the procedure must be modified, it can be done with minimal effect upon data parameters; if data changes occur, the data parameters can be changed without affecting the

LABEL

LESS

LINE COUNT

LINES

LN - Natural logarithm. A mathematical function that may be used in arithmetic expressions. Calculated in floating-point arithmetic.

LOCK - To prevent a tape from being read or written by program control.

LOG - Common Logarithm. A mathematical function that may be used in arithmetic expressions. Calculated in floating point arithmetic.

LS - LESS than. Used in relational expressions.

MAGNETIC - Part of descriptive name, Magnetic Tape Handler.

MASS - Part of descriptive name, Mass Random Access Data storage.

MEMORY - Main storage, core storage.

MODE - A system of data presentation or processing within the information processing system.

MODULE(S) - Refers to core memory size; one module is 4096 words of storage.

MOVE - To transfer a constant, element, field group, record, or array to a constant, element, etc. of the same size.

MULTIPLE

MULTIPLY - To multiply two quantities and store the result in the last-named field or the specified field.

NEGATIVE

NEQ - Not equal to. Used in relational expressions.

NEXT~PROGRAM - An optional Identification Division sentence name.

NGR - Not Greater Than. Used in relational expressions.

NINE(S) - A figurative constant used in procedure sentences.

NLS - Not Less Than. Used in relational expressions.

NO

NOT - May be used in relational expressions. In logical expressions, it is an exclusive negative.

NOTE - To permit the programmer to write explanatory material in the source program for inclusion in the Edited List, but excluded from the compilation.

OBJECT~COMPUTER - An optional Environment Division sentence name.

OBJECT~PROGRAM - Scc Glossary

OF

OMITTED

ON

ONE(S) - A figurative constant used in procedure sentences.

OPEN - To initiate the processing of input and output files. Checks or writes labels and does other input-output functions.

OPTIONAL

OR - A logical operator

OUTPUT - A mandatory Data Division section name.

PAGE

PAPER - Pertaining to High-Speed Printer forms.

PERFORM - To cause the specified section to be executed. Control automatically reverts to sentence following the PERFORM.

PLUG(S) - Refers to connectors on the controller selector to which input-output unit controllers are attached.

POSITION

POSITIVE

PRINTER(S) - Pertaining to High-Speed Printer.

PROCEDURE - A GECOM Division name.

PROCEED

PROGRAM - A complete sequence of data processing instructions. May refer to an object program or a source program.

PROGRAM~ID - A mandatory Identification Division sentence name.

# GECOM PROGRAMMING LANGUAGE

## GENERAL

All compiler programs accept source programs prepared in specialized language and produce an object program ready for computer processing. Unlike most compilers, GECOM is not restricted to an unduly limited acceptable language. The General Compiler language is actually based on several languages.

The GECOM language evolved primarily from two recent major data processing languages, the business-oriented COBOL and the algorithm-oriented ALGOL. Both languages were developed for solving widely different problems, although from the viewpoint of compiler development they have similar characteristics. These similarities made it possible to provide in one complete and compact package a variety of proven programming techniques. COBOL, which satisfies the needs of the broadest spectrum of data processing applications, provided a basic vocabulary (words and symbols), a basic set of rules of grammer or syntax, and punctuation for clarity. ALGOL, to accommodate the demands of scientific applications, contributes Boolean expressions, floating-point arithmetic, and the ability to express equations concisely.

Many computer applications require neither the extensive file processing facilitated by COBOL, nor the profound mathematics that ALGOL provides, but do involve massive numbers of sequential decisions. To cope effectively with these decisions, General Electric devised structure tables for expressing the relationship of decision parameters. These decision structure tables, and the language in which they are expressed, have been termed TABSOL.

TABSOL has been incorporated into the language accepted by the General Compiler and can be used in combination with the COBOL and ALGOL-like capabilities of GECOM.

In addition to file processing, mathematical applications, and complex decision series, much programming effort is and has been devoted to applications involving report generation. The Report Writer format and language, fully compatible with the General Compiler, gives a fully documented method for preparing reports with minimum programming and debugging effort. The Report Writer is an extension of GECOM and derives much of its advantage from the GECOM system.

Both TABSOL and the Report Writer are discussed in the section, "Extensions to GECOM".

GECOM language is not compartmentalized into the component languages discussed above. In a given source program, it is possible to use COBOL statements containing ALGOL-like algebraic notations; TABSOL decision structure tables can be interspersed with procedure statements; and the Report Writer can be used for report generation. The source program can be prepared using one or all facets of the GECOM language. In addition, if the application so requires, GAP coding sequences can be inserted at will.

## COBOL

Because the GECOM language is based primarily on COBOL, some discussion of COBOL and the history of its development is warranted.

In 1959, a meeting was called in the Pentagon by the Department of Defense to consider the desirability and feasibility of establishing a common language for the adaptation of computers to data processing. Representatives from both users and manufacturers were present. The consensus was that the project was definitely both desirable and feasible. As a result, this Conference on Data Systems Languages (CODASYL) established three committees, Short Range, Intermediate Range, and Long Range, to work in four general areas:

Data Description
Procedural Statements
Application Survey
Usage and Experience

In September, 1959, the Short Range Committee submitted a preliminary framework upon which an effective common business language could be built. After acceptance by the Executive Committee of CODASYL, the report was published in April, 1960, by the Government Printing Office as "COBOL-A

WORKING (∼STORAGE) - A mandatory Data Division section name.

WRITE - To display a limited amount of information on the console typewriter.

-To release a record or group to an output file.

ZERO(S) - A figurative constant used in procedure sentences.

ZEROES - SAME as ZERO(S)

Advanced compilers are not limited to accepting simply symbolic instructions, but can accept statements approximating ordinary English sentences or mathematical equations. Most of these compilers are highly restrictive in the vocabulary and syntax permissible and in the equipment that can be used. The GECOM system is the first to utilize a General Compiler program to permit both English-language and algebraic programming and, at the same time, to embody provisions for structured decision tables and automatic report writing. Additionally, the General Compiler has built-in provision to expand its language capability to encompass other source languages yet to be constructed.

Many of the advantages of compiler programs, particularly those associated with the General Compiler are pointed out in the section, ''Advantages of GECOM''. Because the balance of this manual is devoted to describing the GECOM system, it would be redundant to further discuss compilers in general.

However, by virtue of the changing requirements placed upon the programmer who may be engaged in GECOM programming, some consideration should be given to his job title.

~~The average data processing application involves~~ two broad phases. One phase, defining the problem and determining the general method of solution, is generally called systems analysis. The other phase, involving the actual preparation of the program for computer entry, is variously called coding or programming, although in the strict sense coding is only a subordinate part of programming. In some installations, the two phases are performed by separate individuals; in others, both are performed by one person.

The programmer or systems analyst who is thoroughly trained in GECOM principles can communicate more readily with the computer through the General Compiler and, simultaneously, view the overall application in proper perspective. For this reason, the title, systems programmer, is suggested and used in the balance of this manual to describe the GECOM-trained programmer.

# APPENDIX 2. SUMMARY GUIDE FOR GECOM FORM PREPARATION

The following pages briefly summarize the basic rules to be followed in preparing GECOM source programs on the General Compiler Sentence and Data Division Forms. A copy of this appendix is used to provide novice programmers with a convenient guide and a ready reference while becoming familiar with GECOM.

Descriptions of constants are also accepted by assembly programs. Constants, such as the English word TAX or decimal numbers like 365 are accepted by the assembly program and converted automatically into their machine language equivalents. A legend generally accompanies each description of a constant in the source program to indicate what kind of constant is being described. The legend ALF could be used, for example, to indicate alphabetic constants and DEC for decimal constants.

An assembly program producces the machine language versions of constants and instructions in the object program in such a way that they can be loaded into memory at a later time. Generally, a list is also provided, displaying the symbolic descriptions side-by-side with the output produced in the assembly process for each. The list, called an assembly listing, provides an important documentation of the program. It often contains, also, such aids to program checkout as indications of errors in descriptions and lists of symbolic addresses.

The legends, such as ALF and DEC, that are accepted by the assembly program, but do not stand for actual machine operations, are called pseudo-codes, or pseudo-operations. It is common for an assembly program to provide many of these for the programmer to use. Each extends the ability of the assembly program to prepare or document programs.

The symbolic descriptions of instructions, together with the pseudo-operations that are accepted by an assembly program, constitute what is called an assembly language, or a symbolic language. Although there are numerous exceptions, there is generally one output in machine language for each input in assembly language. For this reason, assembling is often considered to be a one-to-one process.

Symbolic language programming using assembly programs, while considerably simpler and faster than machine language programming, is still highly machine-oriented in that the programmer must have a thorough knowledge of machine-language programming. It is common for source programs written for assembly program processing to result in object programs that are as fast and compact as are equivalent programs prepared directly in machine language. Thus, because symbolic language programs are as efficient as machine language programs, symbolic language programming has almost entirely supplanted the machine language as the basic programming media.

Figure 2 illustrates object program preparation, using an assembly process. First, the programmer prepares the source program in symbolic form, using simple mnemonic codes for the desired machine operations and storage of program constants. Second, the source program is converted to a form suitable for machine entry. The most common representations are hole patterns in punched cards or paper tape or bit patterns on magnetic tape. Usually the programmer prepares his instructions on forms from which a keypunch operator can punch the cards or paper tape for direct entry to the computer or, alternately, for conversion to magnetic tape and the input to the computer.

Next, the assembly program is stored in the computer memory and the source program is input to the computer. The computer, under assembly program control, produces the output -- an object program ready for processing.

At any time after assembiy, the object program, now in machine language form, is input to the computer along with data to be processed. The resultant output -- processed data in the form of punched cards, paper or magnetic tape, or printed reports -- is now ready for use external to the computer.

The assembly system available with the GE-225, as previously mentioned, is known as GAP, for General Assembly Program. For further details, refer to the "GE-225 Programming Reference Manual."

## AUTOMATIC CODING LANGUAGE PROGRAMMING

As pointed out above, the assembly program permits an already-skilled programmer to prepare programs with a minimum of errors by eliminating many of the details of program "housekeeping." It also provides a more readable version of machine language, thus reducing the need for extensive annotation of machine coding. However, it does not eliminate the need for computer and machine language knowledge.

The compiler program permits the programmer to take another large step away from machine-oriented programming and toward problem-oriented language programming. Compiler programs place even more of the burden of object program preparation on the computer by permitting the programmer to state the desired operations in sentence form or in equation form, depending upon the application and the compiler program.

Compilers have several advantages over assembly programs. The language of the compiler is easier for the programmer to learn and easier for him to use, as it is more closely related to his problem. The programmer using a compiler usually does not need as intimate a knowledge of the inner workings of the computer as does the assembly programmer. Programming is faster; the time required to obtain a finished, working program is greatly reduced because there is less chance for the programmer to make a mistake and because most normal errors are detected by the compiler.

SUMMARY GUIDE FOR DATA DIVISION FORM PREPARATION

```
                                                                  1        11

DATA DIVISION. Starts in column 8, ends with period. No other entries.
ARRAY SECTION.       }
TRUE~FALSE SECTION.  } Optional sections as required by program. Start in
INTEGER SECTION.     } column 8 and end with a period.
FILE SECTION. Identifies characteristics of data in input and output
              files of the object program. Starts in column 8 and ends
              with a period. Mandatory section.
OUTPUT FILES. Introduces output file descriptions. Starts in column 8
              and ends with period.
INPUT FILES. Introduces input file descriptions. Starts in column 8
             and ends with a period.
WORKING~STORAGE SECTION. Introduces working storage descriptions.
              Starts in column 8 and ends with a period. Mandatory.
COMMON~STORAGE SECTION. }
CONSTANT SECTION.       } Optional sections as required by program.
                          Start in column 8 and end with period.
FD File description. Name follows in columns 11 through 22, 12
              characters or less.
```

are held in the storage element along with the data to be processed. This not only permits step-by-step data manipulation -- it enables the machine to manipulate its own instructions as if they were data. Thus, it is possible for a program to modify itself (if prepared with this intention) and selectively repeat desired portions.

All information processing systems have a repertoire of permissible instructions; these vary in number and scope from one machine type to another and between manufacturers. For any given system, however, instructions can be grouped by general function:

1. Arithmetic
2. Decision
3. Input/Output
4. Control

Arithmetic instructions, as the name implies, enable the data processor to perform arithmetic such as addition, subtraction, multiplication, and division.

Decision instructions enable the system to compare certain data with some standard (other data, perhaps, or the status of some data processor element) and select alternate courses of action.

Input and output instructions permit the reading in and writing out of data via peripheral input/output units.

Miscellaneous control instructions vary most widely between machines and depend largely upon machine design. In general, simpler machines require more control instructions to accomplish a given function or process than do more complex machines.

Even in the most complex machine, individual instructions are very simple operations and a number of them must be used in the proper order to perform a given function.

For many reasons, most modern information processors are designed to operate internally in some form of the binary (two-digit) number system, or a binary-based system, rather than the conventional decimal (ten-digit) system. Certain computer elements are bi-stable devices (that is: conducting or nonconducting, on or off, open or closed) with the two possible conditions expressed as "0" and "1", corresponding to "off" and "on", respectively. The "0" and "1" represent the two digits of the binary number system and are commonly called bits, for binary digits. By grouping computer elements and assigning values to them according to their position in the group, all numbers may be expressed in binary numbers; for example:

9 = 1001     18 = 10010     523 = 1000001011

wherein the 1-bits, by virtue of their position, have values corresponding to the powers of two (1, 2, 4, 8, 16, 32, 64, 128, 256, 512, etc. from right to left). The 0-bits, of course, as in the decimal system, denote zero value and establish position. Thus, the first 1-bit following the equal sign in the example, 9=1001, has a weight of eight (the third power of two), and the rightmost 1-bit has the weight of one (the zero power of two).

A somewhat similar system permits the representation of alphabetic and special symbols in coded binary form. In fact, the system described so briefly here is only one example of many binary numbering schemes in use and is used primarily to show the concept and illustrate the complexity of programming in a pure machine language. It is rarely necessary to program most modern computers directly in binary or machine language form.

As a final example of machine language programming, a simple routine or program for a hypothetical binary computer is used. Assume that two numbers are in the main storage of the computer at locations arbitrarily called 1000 and 1001. It is desired that the two numbers be added and the result be placed in another storage location, 1002. The binary coding for this program might appear as follows:

```
(1) 000000000001111101000
(2) 000010000001111101001
(3) 000110000001111101010
```

The internal computer circuits would interpret such a program thusly:

(1) Load the contents of storage location 1000 into the arithmetic unit.

(2) Add the contents of storage location 1001 to the contents of the arithmetic unit.

(3) Store the new contents of the arithmetic unit in storage location 1002.

Obviously, pure binary programming is slow and tedious, partly because of the difficulty in keeping track of long strings of bits. One innovation that alleviates this difficulty is the use of an intermediate numbering system between the pure binary and the more familiar decimal system.

If the binary numbers in the example above are grouped into three's, as illustrated below, and repetitively assigned the values of the first three

## SUMMARY GUIDE FOR DATA DIVISION FORM PREPARATION (continued)

3 11

F Indicates a field of an input record. Field name is entered in columns 11 through 22.

P Assumes field is packed or unpacked,

U unless it conflicts with a higher level entry (group, record, or file).

1 Assumes one-word binary numeric data. If the data is not integer, a scaling factor must be supplied in the data image columns.

2 Assumes two-word non-standard binary numeric data. If data is not integer, see note above.

S The preceding image is to be used for this entry. Cannot be used if preceding image has a 1 or 2 in column 37.

▨ If any input groups or fields are repeated consecutively, the number of times repeated is entered here.

## THE INFORMATION PROCESSING SYSTEM

Although the effective use of the GECOM system does not require a detailed knowledge of machine-language programming or data processing systems, some such knowledge is desirable, and perhaps is essential if a valid evaluation of the system is to be made.

Data processing needs have resulted in the development of a great variety of computers. While the physical form and the specific logic flow differ widely, general functions and information flow are similar.

The modern computer or information processor consists of five elements as illustrated in Figure 1: Input, Output, Storage, Arithmetic-Logic, and Control. Communication with the computer is possible only through the input and output elements.

The term, input element, is a functional concept, not the name of a unit of equipment. Only through the input element can data enter the processing system. A system may have one or more of several input media: punched cards, punched paper tape, magnetically-encoded tape, or specially-printed documents. Not all computers have available all input media.

The output element makes it possible for the system to perform a useful function; without an output intelligible to the user, a data processor is useless. Output can take one or more of these forms: punched cards, paper tape, magnetic tape, printing, or any of several special-purpose, machine-controlled forms, such as magnetic-ink encoded (MICR) documents.

Input data must be presented to the system in such a way that the system can manipulate and store it internally. For this reason, data is fed into the system in a form that can be readily converted to the internal electronic language of the system (machine language). Similarly, output data is reconverted to an externally-usable form after processing.

The storage element is functionally subdivided into two general types of storage. One, characterized by limited capacity, high speed, and relatively high cost, is referred to as main storage, memory, core storage, core memory, or simply "core". The latter three terms are popular because tiny magnetic cores are the storage medium in many data processors. The other general type of storage, characterized by high capacity, lower speed, and lower cost, is called auxiliary storage. Auxiliary storage may take almost any form, with punched cards and magnetic tape, discs, and drums being the most common.

The arithmetic-logic element contains the circuits that perform the manipulations of data required by the task or application. It adds, subtracts, multiplies, divides, shifts and rearranges data, and makes decisions, according to the purpose of the program. Capabilities vary widely between different types of computers.

The control element decodes and interprets the stored instructions in proper sequence to achieve the purpose of the program.

In a given compter, it can be difficult to recognize physically the separate storage, control, and arithmetic-logic elements. Functionally, they are separate and distinct elements in all data processing systems and should be so considered. The input and output elements are more readily recognized; more often than not they are packaged as separate units, such as card readers, paper tape readers, document handlers, magnetic tape handlers, card punches, paper tape punches, and printers.

## GENERAL PROGRAMMING CONCEPTS

Programming is essentially the framing of a set of directions for a computer. A set of such directions prepared for, and to be communicated to, a computer to guide and control it for a particular processing task is a program.

A subroutine, on the other hand, is a set of directions that is generally incomplete (by itself) in the sense that it usually is only part of a program. Programs frequently contain subroutines for directing the performance of discrete portions of an overall data processing application.

Programs and subroutines, in turn, consist of instructions, which are basic and are the smallest meaningful part of a program. Thus, instructions are the basic tools of the programmer from which he frames the set of directions a computer is to follow.

The phrase "to direct a computer" indicates communication, and communication implies language. In practice, a programmer may use several languages in preparing programs, depending upon the computer. Digital computers are constructed and organized so that they can accept coded representations of letters and numbers, and interpret them as directions to be followed in processing data. Programming languages generally fall into one of three categories, depending on how closely related they are to the computer requirements for accepting information. These three categories are: machine language, symbolic language, and automatic coding language.

### MACHINE LANGUAGE PROGRAMMING

Perhaps the most important characteristics of modern information processors is the stored-program concept. In the information processor, instructions

SUMMARY GUIDE FOR DATA DIVISION FORM PREPARATION (continued)

5 11

FL Field literal. Any legal data name. Used for named fields with fixed values. Rules that apply to fields also apply to field literals. Actual value of literal is enclosed in quotation marks in columns 55 through 80.

OUTPUT RECORD ENTRIES:

R Output record-Name in columns 11 through 22; may be qualified by entry of a qualifier in columns 24 through 35. If record name is unique, It need not be qualified.

P Forces all levels within record to be
U packed (P) or unpacked (U) except binary numerics.

*G *group name in columns 11 through 22. May be qualified. If 2 qualifiers are needed,first goes in columns 24 through 35, second in
~ next line columns 24 through 35 and a tilde in column 7.

P Forces lower levels to be
U packed or unpacked.

# INTRODUCTION

## WHAT IS GECOM?

The GE-225 GECOM system is an advanced and highly effective method for preparing sets of directions for the GE-225 Information Processing System. As a system, it consists of three elements: Language, Compiler, and Computer. These three terms are further explained below.

### THE LANGUAGE

A language is, in general, a means of communication. In the visual form, it usually consists of a set of symbols (such as our alphabet), which can be arranged into meaningful groups (words). Properly arranged aggregates of these groups or words can communicate ideas, action, commands, and questions.

The direction of an automatic information processing system in the performance of a given operation requires communication between man and machine. Just as communication between two men requires a language intelligible to both, communication between man and machine requires a common language. This common language can be machine-oriented (that is, related closely to the basic means by which the computer accepts and presents information, and requiring tedious translation by man of his directions into machine-acceptable form), or the language can be problem-oriented (enabling man to express directions in a form more convenient to the application and placing the burden of the translation on the computer), or it can lie somewhere between these extremes. Machine-oriented and problem-oriented languages are discussed further in the section, ''General Programming Concepts''.

The GECOM language is a problem-oriented language designed to handle scientific problems as well as general business information processing. The primary basis for the language structure is COBOL, the COmmon Business-Oriented Language for programming digital computers. COBOL is further discussed in the section, ''GECOM Programming Language''.

In addition to the capabilities derived from COBOL, GECOM language incorporates many of the features of ALGOL, (an ALGOrithmic Language for stating mathematical computations), such as capabilities to evaluate complex equations, Boolean expressions, and mathematical functions. These computations may be performed in either fixed or floating-point arithmetic.

Further versatility is provided by the incorporation of TABSOL and the Report Writer into the language. TABSOL, for TABular Systems-Oriented Language, is a system for expressing decision logic in a simple tabular form. The Report Writer facilitates report preparation and improves documentation. TABSOL and the Report Writer are discussed in the section, ''Extensions to GECOM''.

GECOM language is not limited to the language capabilities and the extensions mentioned above. General Compiler versatility permits inclusion of GAP, the basic symbolic language (machine-oriented to a degree) of the GE-225 Information Processing System. GAP, for General Assembly Program, is a straightforward symbolic assembly system for the GE-225.

### THE GENERAL COMPILER

If communication with the computer is to occur in problem-oriented language, some means must be provided to translate that language within the computer into machine-oriented form. A set of directions for a computer, regardless of the language in which it is prepared, is called a program or, sometimes, a routine. A program, manually prepared, is generally termed a source program. A source program which has been translated into a machine-oriented program is an object program. One means of translating a source program into an object program is to use a specially-prepared program (called a compiler) which, within the computer, operates upon the source program as if it were data and transforms it into an object program.

The General Compiler (from which the GECOM system derives its name) is a unique program specifically designed to reduce sharply the traditionally high programming costs associated with the computer applications. GECOM is a highly versatile and dynamic ''program generator''; versatile because it accepts source programs written in a variety of languages; dynamic because both the range of languages and the computer types to which it is applicable can

7 11

L Literal; no name used. All other columns are completed as for fields and elements.

OTHER OUTPUT RECORD ENTRIES

▨ Not used for output entries.

▨ B or other character forces lower levels with numeric data description (9) to be in standard binary form unless lower level Format indicates non-standard binary data. A blank in column 43 forces BCD data output.

▨ Forces unpacked data to be left
L (L) justified and zero filled or
R right (R) justified and blank filled.

# ACKNOWLEDGEMENT

"This publication is based in part on the COBOL System developed in 1959 by a committee composed of government users and computer manufacturers. The organizations participating in the original development were:

Air Materiel Command, United States Air Force
Bureau of Standards, Department of Commerce
David Taylor Model Basin, Bureau of Ships, U. S. Navy
Electronic Data Processing Division, Minneapolis-Honeywell Regulator Company
Burroughs Corporation
International Business Machine Corporation
Radio Corporation of America
Sylvania Electric Products, Inc.
Univac Division of Sperry Rand Corporation

In addition to the organizations listed above, the following other organizations participated in the work of the Maintenance Group:

Allstate Insurance Company
Bendix Corporation, Computer Division
Control Data Corporation
DuPont Corporation
General Electric Company
General Motors Corporation
Lockheed Aircraft Corporation
National Cash Register Company
Philco Corporation
Standard Oil Company (N. J.)
United States Steel Corporation

This COBOL-61 manual is the result of contributions made by all of the above-mentioned organizations. No warranty, expressed or implied, is made by any contributor or by the committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or by the committee, in connection therewith.

It is reasonable to assume that a number of improvements and additions will be made to COBOL. Every effort will be made to insure that the improvements and corrections will be made in an orderly fashion, with due recognition of existing users' investments in programming. However, this protection can be positively assured only by individual implementors.

Procedures have been established for the maintenance of COBOL. Inquiries concerning the procedures and methods for proposing changes should be directed to the Executive Committee of the Conference on Data Systems Languages.

. . . . . . . .

Any organization interested in reproducing the COBOL report and initial specifications in whole or in part, using ideas taken from this report or utilizing this report as the basis for an instruction manual or any other purpose is free to do so. However, all such organizations are requested to reproduce this section as part of the introduction to the document. Those using a short passage, as in a book review, are requested to mention "COBOL" in acknowledgment of the source but need not quote the entire section."

SUMMARY GUIDE FOR DATA DIVISION FORM PREPARATION (continued)

9 11.

| | |
|---|---|
| Position contains an alphabetic character, A-Z, or a blank. | A |
| Position contains an integer 0-9. | 9 |
| Position contains a numeral 0-9 with an 11-row overpunch when negative and no overpunch when positive. | R |
| Position contains a numeral 0-9 with a 12-row overpunch when the field is positive and an 11-row overpunch when the field is negative. | I |
| Indicates an assumed decimal point. Neither the V or the decimal point occupy an actual field position. | V |
| Indicates number following E is a power of ten to which the number preceding the E must be raised. E does not occupy field position. | E |

# PREFACE

## ABOUT PROGRAMMING

The programming of information processing systems has traditionally been a costly and time-consuming part of automatic data processing. In the past, many applications that otherwise would readily lend themselves to data processing techniques were avoided because of programming costs. Efforts to improve programming techniques have been directed toward producing faster, more economical, and more accurate programs by placing more of the burden on the data processing equipment.

Various combinations of symbolic coding systems (with one-to-one correlation between machine code and symbolic code), macro-instruction coding systems (with a many-to-one correlation between machine code and macro-code), libraries of standardized subroutines, and other innovations were developed to accelerate programming. Despite these improvements, programmers still prepared programs in terms dictated primarily by the computer; programming languages remained essentially machine-oriented languages.

Today, compiler programs provide the programmer with additional leverage. Program coding can be done in a language more suited to the problem instead of in the purely machine-oriented data processor language.

The GE-225 GECOM system, an advanced and effective automatic coding method, provides the next logical step in programming evolution. GECOM is a step toward fulfillment of the much-needed total systems concept--a concept that deems an information processing system to be an integration of application, programming, and information processor or computer.

The GECOM system is further characterized by its applicability to all classes of information processing problems, its ability to grow, and its inherent provisions for use by future General Electric general-purpose computers. GECOM permits coding in the problem languages of business, science, and industry. GECOM can be adapted to future extensions of existing problem languages as the requirement arises, without obsoleting programs prepared to present specifications.

## ABOUT THIS MANUAL

This manual is presented as a general information manual about the GE-225 GECOM system and is organized to fill the needs of many people having different levels of familiarity with automatic information processing.

For readers with no previous experience in data processing or computer programming, it is suggested that the entire GE manual be covered. Persons having such previous experience, but who are unfamiliar with the GE-225 Information Processing System, are referred to other General Electric publications, listed below.

Readers already familiar with the fundamentals of programming can begin directly with the section, GECOM Programming Language, with no loss in continuity.

Following the section on GECOM programming language is discussion of the Basic GECOM System. All elements are discussed briefly with the intent of providing overall familiarity with all aspects of GECOM.

The next section treats the two major extensions to GECOM, (TABSOL and the Report Writer), which are first mentioned in the GECOM programming language section, but are more effectively discussed after an understanding of GECOM is achieved.

The reader should not assume that reading this manual will make him a master GECOM programmer. The most effective use of GECOM depends upon training and application. More detailed information concerning the various aspects of the GECOM system can be found in the following General Electric publications:

| | |
|---|---|
| GECOM | GE-225 Language Specifications<br>GE-225 General Compiler Operations Manual, CD225H1 |
| TABSOL | GE-225 TABSOL Manual, CPB 147<br>GE-225 Introduction to TABSOL, CPB 147 A |
| GAP | GE-225 Programming Reference Manual, CPB 126 |

SUMMARY GUIDE FOR DATA DIVISION FORM PREPARATION (continued)

| | | | |
|---|---|---|---|
| | | 11 | 11 |
| Inserts a comma in corresponding field positions. Automatically suppressed by floating dollar signs, zero suppression, asterisk filling. | , | | |
| If position occupied by Z in numeric field becomes zero, zero is suppressed and position prints blank. | Z | | |
| If position occupied by * becomes zero, * is printed. | * | | |
| If position occupied by $ in numeric field becomes zero, move $ into it. | $ $ | | |

END PROGRAM. The final entry of the data division must be END PROGRAM starting in column 8 and terminating with a period.

SOFTWARE MANUALS

GENERAL ELECTRIC reserves the right to make alterations, advances, or modifications to the existing program for reasons of increased efficiency.

# APPENDIX 3. SOURCE PROGRAM ORDER FOR COMPILATION

| | | |
|---|---|---|
| I. | IDENTIFICATION DIVISION | Mandatory |
| | PROGRAM~ID. | Mandatory |
| | NEXT~PROGRAM | Optional |
| | AUTHOR. | Optional |
| | DATE~COMPILED. | Optional |
| | INSTALLATION. | Optional |
| | SECURITY. | Optional |
| | REMARKS. | Optional |
| II. | ENVIRONMENT DIVISION. | Mandatory (whether or not any sentences follow) |
| | OBJECT~COMPUTER. | Optional |
| | I~O~CONTROL. | Optional |
| | FILE~CONTROL. | Optional |
| | COMPUTATION~MODE. | Optional |
| III. | PROCEDURE DIVISION. | Mandatory |
| | Closed sections and decision tables delimited by BEGIN-END | Placement mandatory if sections are used. |
| | Master program | Mandatory |
| IV. | DATA DIVISION. | Mandatory |
| | ARRAY SECTION. | Optional |
| | TRUE~FALSE SECTION. | Optional |
| | INTEGER SECTION. | Optional |
| | FILE SECTION. | Mandatory* |
| | OUTPUT FILES. | Mandatory* |
| | INPUT FILES. | Mandatory* |
| | WORKING~STORAGE SECTION. | Mandatory* |
| | COMMON~STORAGE SECTION. | Optional |
| | CONSTANT SECTION. | Optional |
| | END PROGRAM. | Mandatory* |

* The section heading card is mandatory; further entries under it are optional.

# ILLUSTRATIONS

# APPENDIX 4. GLOSSARY

A list of important terms (most of which are used frequently in the body of this manual and many of which are encountered frequently in other GECOM literature) have been included in this glossary. Most definitions are deliberately brief and are not intended to be comprehensive; many of the terms have additional meanings. For more detailed and more exhaustive listings, the reader is referred to any of several excellent glossaries of information processing terminology.

ADDRESS - A specific location in storage or memory. Actual addresses are numeric. Addresses used in GECOM are symbolic, that is, represented by names.

ARITHMETIC EXPRESSION - A sequence of data names, numeric literals, and/or mathematical functions connected by mathematical symbols.

BCD - Binary Coded Decimal; a system for representing any character of the character set of the computer by a group of binary digits.

BEGINNING FILE LABEL - A group of records (blocks) which identifies a file in a multifile magnetic tape. It is block 0, the first block of each file.

BINARY NUMERIC - A digit or group of characters or symbols representing the total units using the base two: a number expressed in binary digits or bits, 0 and 1.

BLOCK - A group of records read from or written on magnetic tape as a single physical tape record.

BLOCK SIZE - The number of words in a block.

BUFFER - Storage locations used to compensate for differences in rate of data flow when transmitting data from one device to another.

CHARACTER - One of a set of basic symbols used to express data. Includes decimal digits 0 through 9, the letters A through Z, punctuation, and special symbols.

CONDITIONAL EXPRESSION - An expression that can be either true or false.

CONDITIONAL NAME - A name assigned to a possible value of a numeric or alphanumeric field or element. A conditional name must be described in the Data Division.

CONSTANT - A value used in a program without alteration. Constants are either literal, figurative, or numeric in GECOM.

DATA IMAGE - The characteristics of a data field; that is, length, content, sign, and character type for each position. The data image is used within the Data Division to define data input and output.

DATA NAME - A programmer-assigned word naming a file, record, field, constant, or other data. Data names are composed of letters, numerals, and hyphens, not exceeding 12 characters, and may be names of records, groups, fields, arrays, elements, sections, or true-false variables.

ELEMENT - A subdivision of a field. For example, a date field could contain a DAY element, a MONTH element and a YEAR element.

FIELD - A unit of data within a record. It may or may not be a part of a group.

FIGURATIVE CONSTANT - A special name representing specific values [ZERO(S), ZEROES, SPACES, ONE(S), through NINE(S)]. May be used in procedure sentences to imply strings of characters.

FILE - A set of records

FIXED-POINT - A number which includes a decimal point, either between digits or following them (1.23, 123., or 123.0)

FLOATING-POINT - A number expressed as a whole number, a decimal fraction, and a power of ten. (1.287*$10^{-2}$)

GENERATED FIELD - A field (of data) which is generated as a result of calculations and is not input to the program.

INSTRUCTION - A group of symbols causing the data processor to perform some operation.

INTEGER (as used in this manual) - A number of 5 digits or less not containing a decimal point.

# CONTENTS

*Progress Is Our Most Important Product*

GENERAL ELECTRIC

COMPUTER DEPARTMENT • PHOENIX, ARIZONA

LITHO IN USA