1. INTRODUCTION

1.1. GENERAL

This manual describes the ALGOL language for the UNIVAC 1108 System. The basis for this language is the "Revised Report on the Algorithmic Language, ALGOL 60" (Communications of the ACM, Vol. 6, January 1963, 1–17). This implementation of ALGOL is very close to that of the report. Its one significant omission is the omission of dynamic own arrays. Some of its more significant additions include three new data types (**STRING, COMPLEX, REAL 2**), and default declarations. Provision is made for inclusion of procedures written in assembly language or FORTRAN V.

This manual is intended as an introduction to ALGOL 60 and as a reference manual in the use of UNIVAC 1108 ALGOL and is not intended as an exhaustive, self-contained description of ALGOL 60. The text consists principally of definitions and rules for writing ALGOL programs, examples of these rules, and some sample programs.

A set of appendices includes special sections on file-handling procedures, UNIVAC 1108 ALGOL syntax in chart form, and sample ALGOL programs; lists of basic symbols, library procedures, and diagnostic messages.

1.2. THE ALGOL COMPILER

The ALGOL compiler is a program which accepts statements expressed in ALGOL and produces programs for the UNIVAC 1108 System.

An ALGOL program is a sequence of statements written in ALGOL language. These are translated by the compiler into the language of the computer: machine language. The ALGOL statements are called the source code, and the translated statements are called the object code. The compiler itself is a program written in machine language and is called the UNIVAC 1108 ALGOL 60 Compiler. While translating the ALGOL statements, the compiler looks for errors in syntax (that is, for errors in the forms or construction of statements).

The compiler operates in two passes. The first pass scans the statements and does about 95 percent of the work required to produce the final object code. The second pass goes into operation immediately after all the statements have been scanned; it completes the remaining details of producing the object code. Upon successful compilation, the object code can be read into the main storage and executed. Activities that occur during compilation are sometimes referred to as compile-time activities; for instance, compile-time diagnostics. The execution phase is referred to as run-time.

1.3. ALGOL 60 AND UNIVAC 1108 ALGOL

There are several differences between ALGOL 60 as defined in the revised report and 1108 ALGOL 60. In that ALGOL is intended as a standard language and compatibility of programs between machines is becoming more and more important, those differences must be explicitly pointed out. They fall into two classes: extensions to ALGOL 60 and definition of things left undefined by the report; modifications or omission of ALGOL 60 entities.

1.3.1. Extensions to ALGOL 60

Extensions to ALGOL 60 include the following:

- **STRING** and **STRING ARRAY** variables enhance the value of ALGOL as a data processing language.
- New arithmetic types **COMPLEX** and **REAL 2** enhance the value of ALGOL to scientific users.
- **XOR**, an additional Boolean operator is provided.
- **EXTERNAL PROCEDURE** declarations are provided for convenience in programming large problems and for building libraries.
- I/O and other library procedures are provided and, related to them, are the **FORMAT** and **LIST** declarations.
- A compact form for **GO TO** and **FOR** statements is allowed.
- Variables are given values of zero or blank at the entrance to a block; thus initialization statements need not be made.
- The controlled variable of a **FOR** statement has a defined value when the statement is terminated by exhaustion of the **FOR** list.
- The **OTHERWISE** declaration or declaration by default is provided.
- The variables in a multiple assignment statement need not be the same type.
- Arguments of type **COMPLEX** and **REAL 2** are permitted for various standard functions.

1.3.2. Deviations from ALGOL 60

- Because of hardware requirements, identifiers are unique only to their first twelve characters and may contain no blanks; numbers may contain no blanks, and certain basic symbols are reserved identifiers (see Appendix A).
- **OWN** arrays are not dynamic.
- Numeric labels are not allowed.
- The comma is the only parameter delimiter allowed in a procedure call.
- A **LOCAL** declaration is required to resolve all forward references to identifiers.
- An integer raised to an integer power always produces a **REAL** value.

- All the formal parameters of a procedure must be specified and must agree in type with the actual parameters.

These and other restrictions are covered in more detail in later sections of this manual.

1.4. LANGUAGE CONVENTIONS

ALGOL is described in terms of three languages in this manual: reference, publication, and hardware language.

The reference language is that which defines ALGOL in the ALGOL 60 Revised Report. It is computer independent and utilizes the basic ALGOL symbols to define the language syntax and semantics. Throughout the text, but sparingly, the syntax of 1108 ALGOL is described in terms of this reference language.

Example:

<identifier> : := <letter> / <identifier><letter> / <identifier><digit>

(Read : := as 'is' and / as 'or')

This says that an <identifier> is either a <letter> or an <identifier> followed by a <letter> or an <identifier> followed by a <digit>. Further discussion of identifiers is found in 2.2.

While having the advantage of compactness and precision, the formalism is not suitable as an introduction to ALGOL and so has been used only as a summary aid. Except for the formal definitions of the reference language, the hardware language (the language acceptable to the UNIVAC 1108) has been used throughout the text. All basic symbols which appear in the text, as well as all examples, are written in upper case letters. This is the form in which they appear in the hardware language.

For publication purposes, the boldface type delineates the basic UNIVAC 1108 ALGOL symbols. Transliteration rules for basic symbols are given in Appendix A.

Statements may be separated from each other by either the semicolon or the dollar sign. Because of keypunch limitations, the $ is commonly accepted and has been used in all examples throughout this manual.

The following symbols are considered equivalent:

'' is equivalent to : (colon)
= is equivalent to := (replacement operator)