X-564-66-73

NASA TM X-55538

PROGRAM DOCUMENTATION

BY

HOWARD R. STAGNER

FEBRUARY 1966

GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

X-564-66-73

PROGRAM DOCUMENTATION

by

Howard R. Stagner

February 1966

Goddard Space Flight Center
Greenbelt, Maryland

PROGRAM DOCUMENTATION

by

Howard R. Stagner
Goddard Space Flight Center
Greenbelt, Maryland

30361

ABSTRACT

A discussion of programming documentation, the need for it, its functional role, and structure is given, followed by a detailed outline of requirements for complete documentation.

TABLE OF CONTENTS

PROGRAM DOCUMENTATION

INTRODUCTION

Program documentation, particularly from a program maintenance standpoint, has traditionally received the lowest priority and been the least adequate feature of any programming task. The two prime reasons for this are, of course, lack of time and lack of standards. Coming last in the chronological process of designing, coding, and checking out a program, there is usually not enough time to adequately prepare the final documentation of a program before the money allocated for development of that program runs out or the programmer has to go on to some other tasks. Usually what happens is the documentation that grew along with the program is hurriedly combined with the original specifications (which are no longer completely valid by this stage of the development), and then minimum operating instructions are added to form the final documentation. A major fallacy of this procedure is that what may have been adequate documentation during development of a program is usually quite inadequate for later maintenance of that program, which after all is one of the most important reasons for documentation. To make matters worse, programmers themselves are also in a state of flux. A programmer may be here today but tomorrow he will either be with another employer or on a new task with the same employer. In either case he cannot be expected to pick up a program he wrote 6 months ago and readily recall how or even at times what he was doing at certain points in the program, unless he has adequate documentation. In fact I suspect one reason programmers are so transient is to free themselves of the albatrosses around their necks—the old programs that come back to haunt them because a change is needed but nobody else wants to or can make the change because of inadequate documentation.

Certain universal documentation standards do exist, principally in the form of flow chart symbol standards, but by themselves flow charts are inadequate.

This is not to say, however, that there should be an all inclusive documentation standard. Such a standard would be as impractical and unwieldly as a universal program. Documentation requirements obviously vary with the size and complexity of the program, with the experience of the creators, users, and maintainers of the program, and with the use to which the program will be put. Certainly where an agency contracts out of house for the development of a program which will be used and probably maintained inhouse there is a need for rather detailed documentation. The remainder of this paper is an attempt to outline a documentation standard for use in such cases. Depending on the situation, all or parts of this standard may be useful in a particular programming task.

FUNCTIONAL ROLE AND STRUCTURE OF DOCUMENTATION

Final documentation of a program should provide three things: (1) a systems description, (2) operating instructions, and (3) detailed design and maintenance information. In larger systems these functions should be handled in at least three separate documents. The systems description document would be slanted toward the person who is not directly involved with the program but who needs to know what the program does, what environment it operates in, and how it relates to other areas of a larger system. This document would thus be general and tutorial in nature. It should briefly describe the project or overall system of which the program is a part and it should show the relation of this program to the total system, the source of inputs, the purposes and forms of outputs, and the functions performed in the program. This functional description is best presented as a combination of text and general flow diagrams both of the major functional or logical flow type and of the data flow type.

A second document would be directed towards the actual users of the program—the computer operators and those who prepare the input data and handle intermediate and final output data. This document would describe the machine environment required by the program and delineate the format of all program input and output. It should contain a special section for the computer operators in which all set-up requirements, program-operator messages, error handling and recovery procedures, and output labeling information is assembled.

The third document should provide sufficient information for understanding the detailed design and functional behavior of the program so that output results can be verified through an understanding of the specific algorithims used to generate those results and so that the program can be modified to meet changing requirements. Of necessity this document must be considerably more detailed and specific than the first document. It should contain a detailed functional flow diagram with an accompanying description showing all logical paths and the hierarchy and inter-relation of all program elements. All program elements must be fully documented and the internal structure and purpose of buffers, data areas, tables, communication cells, and program switches must be given. Besides describing <u>what</u> is done at all steps in the program, the documentation should describe <u>how</u> it is done and very importantly, <u>why</u> it is done at that particular time and place in the program—without belaboring the obvious.

These three documents may of course be combined into one for smaller programs and may need to be expanded into more modules as the system size increases. In addition the functional arrangement will have to vary to meet the needs of a particular program. For example in documenting a monitor system or source language the second document would be slanted toward programmers and not computer operators. It goes without saying that this basic concept is modular upward and downward to document the assembly of subroutines and subprograms into programs, programs into modules, and modules into large systems.

DOCUMENTATION SPECIFICATIONS

The following categorical requirements should be met for a program to be considered as fully documented. The categories are arranged in a logical sequence such as might be used in documenting a medium size program that stands by itself, however the physical arrangement of the documentation is of necessity dependent on the purpose, type, complexity, and usage of the program and its relation to other programs comprising a complete system.

I. Introduction

A. Purpose of Program—Why the program was developed, what it does in general, and, if applicable, how it relates as part of a system.

B. Environment—Machine requirements of the program in general (machine type, core storage needed, special pherepherals), operating mode (real time, on-line, etc), and types and sources of data input and output.

C. Specifications—Outline the design, production, and operational requirements and restraints of the program, i.e. what it is supposed to do in detail, and when, where, and how it is supposed to do it in general.

II. General Description

A. General logical flow or block diagram of the total program on a major functional or operation level to stress and clarify the ideal flow and the interrelationships between major functions.

B. Description of general logical flow diagram with a key for relating the description to the flow diagram. For each block in the flow diagram the description should explain what is done and why it is done when this is not immediately self-explanatory from the flow diagram. It should mention the techniques employed and give insight into the wherefore of the flow diagram.

C. General data flow diagram of the program on a conceptual level. This may require two diagrams, one showing the external data flow, i.e. tape and cards in, intermediate tape out, final tape out, listing out, and one showing data flow internally to the machine, i.e. tape to core to drum on file basis, back to core on record basis, merging, processing, transferring to output area, etc.

D. Description of data flow diagrams keyed to the flow diagrams. Expository remarks to clarify flow diagrams such as chronological flow, purpose of assembling data on drum temporarily, etc.

III. Detailed Description

A. Programming Standards—Flow chart symbols (if not ASA standards reference on appendix for detailed description), conventions followed, ordering of listings (alphabetical, by function, etc.), internal ordering of program coding (entry points, equates, dimensions, exemtable instructions, tables, etc.) symbolism convention, source languages used, etc.

B. Main Program, Control Program, Driver Program or Main Subprogram

1. Purpose—Short description of what it does, general explanatory information essential to understanding purpose and function of the program

2. Type—Functional type (real-time, driver), source language.

3. External References—Common areas and communication cells that are referenced or set by this routine.

4. Method and Structure—Detailed description of algorithms used, formulas, techniques, causes of error conditions and resulting action.

5. Detailed Flow Diagram—Still on a functional level not representing coding details (summarizing and explaining not echoing coding), yet going into more specific details than the general flow diagram. All logical paths should be shown including paths followed in error conditions. Setting and testing of all program switches should be shown. See also Section III-C-7.

6. Description of detailed flow diagram keyed to flow diagram—should explain what and why for areas in flow diagram that are not immediately self evident and to clarify flow, cause and effect, and interrelations between areas of the program.

7. May contain local symbol and switch glossary, symbol cross reference table, subroutine cross reference table, formats of buffers, tables, list of program flags and their meanings.

C. Subroutines—Each subroutine description should contain the following sections (requirement may be relaxed where section does not apply or in very small subroutines which can be described in a few words).

1. Purpose—Same as detail for main program (Section III-B-1).

2. Calling Sequence—List of parameters, all possible values of parameters and their meaning, format of parameters (floating point, etc.), type of calling sequence (Fortran IV address list, values in arithmetic registers, etc.).

3. Environment—Condition of computer at entrance, during operation, and at exit of subroutine (interrupts disabled, enabled), pre-requisite conditions, register status at entrance and exit, necessary global input data areas, output or result data areas, usage of routine, etc.

4. Type—Source language, mode of operation (realtime, interrupt, normal, re-entrant, recursive, multiprogrammed, etc.).

5. Conventions Used—If local to this subroutine and not described in Section III-A.

6. Method and Structure—Detailed description of algorithms used, formulas, techniques, causes of error conditions and resulting action, other essential information.

7. Detailed Flow Diagram—On functional level not a mirror-image of the coding yet showing all logic paths, the setting and listing of all switches, all decision points, etc. Ideally flow diagram should be such that non-functional changes in the coding affecting only coding techniques, indexing methods, minor instruction sequence should not require a change in the flow chart. See also Section III-B-5.

8. Description of Detailed Flow Diagram—Keyed to flow diagram. See Section III-B-6.

9. External References—Common areas and communication cells that are referenced or set by the routine.

10. Buffers, Tables, and Constants—Formats of local buffers, tables, and constants.

11. Usage for non-general routines to describe what programs use this routine and what tables and communication cells they set.

12. May contain local symbol and switch glossary, local symbol cross reference table, local subroutine cross reference table, micro flow charts, macros local to this routine, etc.

D. Macros, Procedures, Generative Coding

1. Purpose and Function—What it does, general explanatory information essential to understanding purpose and function of the element.

2. Calling Sequence—Names of entry points and parameter string with structure and meaning.

3. Method and Structure—Techniques used, amount of code generated (may vary), internal logics and conditional situations, etc.

E. Program Description Tables

1. Routine Cross Reference Table—Program, subprogram, subroutine, or other routine name, names of routines that call this routine, names of routines this routine calls.

2. Symbol Glossary—Glossary of control words, switches, labels, etc. used in flow charts and coding with definition and explanation of meaning and use of the symbol. Descriptions of control words, switches, communication cells should include a list of all values that switch can assume and the corresponding meaning.

3. Grouping of Common or other globally defined areas and symbols. Listing them in one place eliminates the duplication that would result from listing these items in the description of each subroutine that references them.

F. Input Data Formats

1. The required physical arrangement of all input data must be given. In general the following should be covered in describing individual fields:

a. Usage—Purpose of field, range of possible values and meanings, labels used by program in referencing field.

b. Units—Dollars, milliseconds, engineering units.

c. Type—Fixed field, free field.

d. Format—Field size, location in record, block, and/or file, field delimiting symbols, free field characteristics, sequence of fields, reference to standard formats used such as Fortran types, partial word structure, etc.

e. Representation—BCD, binary, integer, floating point, fixed point, alphanumeric, boolean, etc.

f. Limitations—Requirements for leading zeros, techniques for indicating void or empty fields, signing requirements.

g. Sequence—Sequence between fields, physical blocks (records, cards), continuation symbols, terminating symbols, number of cards).

2. Card Input—List by function, i.e. control card, data card. Give punching standard (BCD code, row or column binary), reference punch code used (IBM, Univac, CDC) all of section 1 above applies. Give card size (80 column, 90 column). Include chart of card arrangement. Describe nature of checksums or parity punches.

3. Paper Tape Input—Number of channels, punch code used.

4. Magnetic Tape Input—Density, parity, channels per character, nature of checksums used. Describe record or block structure, file structure, and tape structure, i.e. multi-file reels, multi-reel files, end-of-file marker usage, end of data on tape indication (double end of file, special block), maximum record size and whether fixed or variable, intermixing of records or blocks if several different types are intermixed, i.e. label record, data record, orbit profile record, file summary record, etc. Give chart explaining above record, file, and tape structures. All of section 1 above applies.

5. Console or Typewriter input all of section 1 above applies.

6. Disk pack or other random access input all of section 1 applies with special emphasis on structure and arrangement of data blocks.

G. Output Data Formats—Same as for input data formats with following additions:

1. Magnetic tape Output—Same as input only also describe handling of end-of-tape marker (tape backspaced, 2 end-of-files written, then tapes swapped or simply swapped with no end of files or backspacing or 2 end of files written after end of tape marks, etc.).

2. Console or Typewriter Output—Can be described in operating procedures section.

3. Printer Output—Section 1 applies. Heading and all fields should be explained.

4. Disk pack or other random access output all of section 1 applies. Emphasis should be placed on fields describing amount of disk used, starting, ending address, data structure.

H. Internal Table and Buffer Formats—Adequate documentation of internal tables and buffers is especially important because this is nothing intrinsic in the source language coding of such non-executable program elements that reveals what they do. For this reason a detailed description of all tables and buffers must be given. This description should cover the following:

1. Name of the Element

2. Purpose or Function of the Element

3. Usage—What routines reference this element, i.e. which ones set values into the element or change values in the element, which ones only reference it, etc.

4. Type of Element—Common area, input buffer, etc.

5. Size of Element

6. Format of the element including a chart and/or listing of the location of each item. In general most of section F-1 applies.

7. Limitations—Range of values that each element can legally assume and the logical action each value governs.

IV. Operating Procedure

A. Program Description—Brief summary of the purpose and functions performed by the program.

B. Computer Environment—Describe computer type, core memory size needed, pheripheral devices and channels required, etc.

C. Set-Up Requirements—Describe complete run deck required to execute the program, giving illustration of sequence of cards and format of cards. Describe input and output assignments and limiting conditions - i.e. requires two blank tapes to be assigned to output logical unit OUT, requires special version of MONITOR, etc. For magnetic tapes give label assignments, number of servos to assign to each label, channels recommended for each label (and channel and unit required if absolute assignment is used). For printer give paper size, number of carbon copies required, etc. Give jump switch settings and their meanings. Describe all parameter cards and all possible values of the parameters and their respective meaning.

D. Execution—Describe any unique requirements for loading and starting execution of the program and describe standard operating procedures for this program. SOP would include action to follow on tape faults, conditions requiring post mortem dumps, etc.

E. Operation—Describe the operation of the program from the computer operators standpoint. This description should supplement the general descriptions given in Section II and Section IV-A. The description should be related to things the operator can observe. For example "following set up instructions the program reads the label records from tapes ALPHA and GAMMA and writes a file heading on the printer. It then merges records from ALPHA and GAMMA, occasionally entering what appears to be a small loop while in computes eigen values. . . . " Only the normal program and data flow should be described here. Error recovery procedures described in section below can then be related to this description.

F. Console Messages

1. Requiring Operator Reply—List alphabetically all program messages which require an action of some sort by the operator. List the action required or the input message options along with a brief explanation of the purpose and resulting action of the message.

2. Not Requiring Operator Reply—List all messages of an informative nature with a more detailed explanation of the cause and meaning of the message if it is not self explanatory.

G. Normal Termination Procedure

1. Describe normal ways that the job can be terminated. List all materials which must be saved and describe how these should be labeled physically and what type of run logging information should be recorded.

H. Error Termination Procedures—List conditions which operator can identify and error termination. Describe all deviations from the normal termination procedure described above such as which output need not be saved, whether or not the operator can recover the run by correcting input cards, and if he can, what specific action he should take to correct the error or what indications such as program register contents he should note on the log sheet for the run.

I. Recovery and Restart Procedures—Describe any normal optional methods available for terminating the run so that it can be continued at a later time (if possible). Describe restarting procedures and recovery procedures from error terminations including methods of validating proper recovery.

V. Program Material

The following physical program material must be furnished as a minimum:

1. Two copies of the final object program, either on magnetic tape or on cards.

2. Two copies on magnetic tape or cards of the source language program from which the final object program was generated.

3. Two copies of the listing of the final source language program bound between hard covers.

4. Two copies of a listing of the contents of a program tape if such a tape is furnished. This table of contents should list program name, version, etc.

5. Two copies of a program memory allocation created at program execution time should be furnished.

6. Samples of typical printer output from the program may be included.

7. Two complete execution decks must be provided.

8. Two complete program update decks (minus the change and update cards) must be provided if techniques other than standard ones are necessary for updating the program.

VI. Appendixes

In editing the documentation liberal use of appendixes can aid in the utility of the document. Appendixes may include such things as flow chart standards, symbol tables, mathematical expansions, and other large tables, discussions, and charts that would tend to break up the flow of the documentation if they were inserted directly in the text.