Standard S24.806
24 January 1986
(Revised 4/19/94)

SOFTWARE DEVELOPMENT, MAINTENANCE, AND USER DOCUMENTATION

1.0 SOFTWARE DEVELOPMENT

1.1 SCOPE

1.1.1 Scope. This document establishes standards to be observed by the contractor in the development of software.

1.1.2 Application. The requirements of this standard shall apply to all software custom developed, or modified, to meet government specifications. Unless otherwise specified, they shall not apply to multipurpose commercially produced production software, supplied with supporting documentation and updates.

1.1.3 Contracting Officer's Technical Representative. The Contracting Officer's Technical Representative (COTR) shall provide the final interpretation of any conflict between this standard and specific contract requirements.

1.1.4 Waivers. Any request for waiver of specific requirements of this standard shall be submitted in writing to the COTR and to the Contract Administrator. A request for waiver must include: a) identification of the paragraphs for which the waiver is requested; b) identification of the systems, equipment, or components for which the waiver is requested; and c) a discussion of rationale for granting the waiver, including impact on reliability, maintainability, schedule, and cost if the waiver is not granted.

1.2 APPLICABLE DOCUMENTS

DOD-STD-2167
"Defense System Software Development"

Military specifications and standards are available from: Command Officer, U.S. Naval Supply Center, 5801 Tabor Avenue, Philadelphia, Pennsylvania 19120.
FIPS-PUB-38
"Guidelines for Documentation of Computer Programs and Automated Data Systems"

Federal Information Processing Standards Publications are available from: The Suprintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

NOAA/NESDIS
Standard No. S24.801 "Preparation of Operation and Maintenance Manuals"

NOAA standards are available from: U.S. Department of Commerce, National Oceanic and Atmospheric Administration, National Environmental Satellite, Data, and Information Service, OSD3, Washington, D.C. 20233.

1.3 PROGRAM STRUCTURE AND FLOW

1.3.1 Modularity. All systems shall be made up of program and subprogram modules, sometimes referred to as routines, and data modules which are linked together through external references.

1.3.1.1 Single Function. Each program or subprogram provides a single distinct function. For example, the main program is usually responsible, along with the interrupt system, for the control function.

1.3.1.2 Size. No limits are placed on size. But as a guideline most of the routines should include no more than one page, approximately fifty lines, of code. Only one instruction statement per line is permitted.

1.3.1.3 Code and Data Separation. All executable code, bit pattern equates, read only data, etc. shall be write protected by hardware or software provided with the system. Resident code and data areas are distinct, and each contains the applicable explanatory text. Read only data should be segregated from read-write data. Local variables within different modules shall not share the same storage locations.

1.3.2 Calling Conventions. A set of calling and return conventions provide the execution linkages between modules.

1.3.2.1 Call and Return Macros. All assembly routines will be referenced for execution via the same call and return macros. The call and return procedures shall include saving and restoring of all systems registers altered, except for those serving to retain results. Routines shall employ macros from the system macro library when available.

1.3.2.2 Single Entry. A single entry point will be used in each routine.

1.3.2.3 Single Exit. A single exit point will be used in each routine, except for error exits.

1.3.2.4 Program Flow. Routines will be organized on the listing so they flow down the page. Branches shall be avoided when using a modular language. This constraint, along with the use of common exits within a routine for each logic section, provide the necessary structure for maintaining program validity and ease of update.

1.3.2.5 Real Time Response Requirements. If real time response constraints dictate faster execution requirements , exceptions to the above conventions (1.3.2.1 through 1.3.2.4) shall be requested as described in 1.1.4. Requirements for using interrupt inhibits shall be clearly described. If a waiver is granted, such occurrence(s) shall be clearly delineated with inline comments, and shall not violate structured code requirements.

1.3.2.6 Argument Passing. Argument passing techniques must follow those used by the equipment manufacturer or other commercial systems software vendor. The use of registers and/or the stack should follow a consistent pattern.

1.3.2.7 Success/Failure Indications. Success/failure indications should follow those used by the equipment manufacturer or other commercial systems software vendor for the same equipment. This is to assure the correct status of such indicators will be maintained throughout the code.

1.3.2.8 Argument Validity. Modules are responsible for verifying the validity of arguments passed to them, including values input from sources external to the system. Test code needed for this function may be removed, if not required by the completed product.

1.3.2.9 Error Conditions. All error condition logic paths must be completed. Error return values cannot be ignored.

1.3.2.10 Operations and Expressions. Mixed mode operations shall be avoided (e.g., arithmetic between real numbers and integer numbers). Compound negative Boolean expressions shall be prohibited. The order of evaluation for compound expressions shall be clarified through the use of parentheses and spacing. Nesting beyond five levels shall be prohibited.

1.3.2.11 Operator Interface Conventions. Operator response requirements and displays to operators shall be consistent and uniform of throughout a system in order to minimize training and reduce the likelihood of operator errors.

1.3.2.11.1 Invalid Response. The software shall reprompt the operator in response to an invalid response. The reprompt may include additional response options (e.g., an abort to the system level). An invalid operator response shall not automatically abort a routine. Exception to this requirement is allowed for system security (e.g., entry, passwords, ect.) conventions.

1.3.2.11.2 Default Response. The default response to a prompt, that is the response assumed when the

`Return' key is struck, shall be: a) the affirmative value; or b) retaining the present numeric value. Re-entry of the current or existing value shall not be required by the operator to retain that value.

1.3.2.11.3. "Y/N" Response. Prompts which can be answered by a `yes'/`no' response shall accept `Y'/`Yes' or `N'/`No,' in either upper or lower case. Prompts requiring `T'/`True' or `F'/`False,' or "1"/"0" shall not be allowed.

1.3.2.11.4 CRT Displays. CRT displays from application levels shall have labels to the left, left justified, and values to the right, right justified. Multiple column formats may be used. In the absence of operator selectable options, all CRT characters shall be at the same intensity. If brightness selections are provided, they shall include: a) values brighter than labels; b) labels brighter than values; and c) uniform brightness.

1.4 SOURCE PROGRAM FORMATTING

1.4.1 Line Format.

Lines within all modules shall be consistent in format for both executable code and comments. Paragraphing, blocking by blank lines, and indentions shall be used to enhance readability.

1.4.1.1 Line Size. All source lines shall consist of from one to eighty characters (not including trailing blanks, carriage returns, and other control characters). Use of a continuation statement is implied by this requirement.

1.4.1.2 Instruction Statements. Instruction lines must be formatted, consistent with the language being used, with respect to the starting column of each field. Only one instruction shall be coded on a line.

1.4.1.3 Comment Statements. Comments should be consistent with respect to starting columns. Indentions shall be used for readability.

1.4.2 Inline Comments.

Inline comments may occur on the same line as an instruction and/or on separate lines.

1.4.2.1 Comment Formats. Comment lines and paragraphs shall be set off from the executable source code in a uniform manner. There shall be no doubt or ambiguity as to what are comments and what is executable code. Separate comment lines and paragraphs shall be preceded and followed by a single blank line. Comments inserted on instruction statement lines shall conform to 1.4.1.3.

1.4.2.2 Comment Content. All coding must be commented throughout to convey the global role of the instruction statements. In assembly language code this usually results in one comment per line of code with paragraphs of comments included for explanation of particularly important or complex sections.

1.4.3 Naming Conventions. Strict naming conventions are to be maintained throughout each system.

1.4.3.1 Global vs. Local Symbols. There must be different naming conventions used for local and global symbols. This is usually a single special character used as a prefix or suffix to the root name.

1.4.3.2 Interrupt Processing. Testing and altering interrupt priorities must be accomplished using a consistent set of symbols, as determined by the language.

1.4.3.3 Name Derivations. Program, data module, constant, and variable names shall be derived so as to meaningfully and clearly identify the function performed or being described. This derivation shall be uniform throughout the system. Language keywords shall not be used as names.

1.4.4 Systems Messages and Codes. A coding scheme must be used for each message within each category of system message (e.g., error, diagnostic, input/output, ...).

1.4.4.1 Codes and/or flags must be set within the routine detecting the abnormal condition.

1.4.4.2 Messages, formatted for screen display, must be made available for each system status condition. These messages should have corresponding code and/or flag settings.

1.4.4.3 All error and diagnostic messages shall be presented in a uniform manner and shall be self-explanatory. They shall not require the operator to perform table look-ups or futher processing of any kind to interpret the message. The message shall indicate the source or cause of the error condition, and, if possible, shall identify both the routine detecting the error the routine causing the error.

1.4.5 Module Preface. Every program and data module must have a preface comment block containing the following information and identifiers. The identifiers must be uniquely recognizable by position and keyword.

NAME	The name given to the module.
TYPE	Data, Main Program, Subroutine, Function, Macro, etc.
RELEASE	The release or base level of the program module.
VERSION	The revision/update number of the release.

MODIFICATION LEVEL	A number indicating the modifications which have been applied to the version.
MODIFICATION DESCRIPTION	A brief statement indication the purpose and function of the modification.
ORIGINATOR-DATE	The name of each modifying author and the date of the modification. Names and dates are entered one per line in chronological order, newest first (at the top of the listing).
FUNCTION	A brief statement of the function of the module.
SYMBOLS	A list of the constants and the definitions of each.
MACROS	A list of the local macro and function definitions.
DATA DECLARATION	This shall include, for both local and global data: a description of each element (type, size, etc.); organization (functional, alpha, etc.); and adjacency requirements (e.g., array elements, common buffers, etc.).
DESCRIPTION	A comprehensive summary of the function of the module.
CALLING SEQUENCE	The precise syntax used when invoking the program.
CALLING ROUTINES	A list of all routines calling (using) the module.
CALLED ROUTINES	A list of all routines called (used) by the module.
INPUTS	A list of the inputs expected by the module; catagorized by the sources; and indicating whether the input is a value or reference parameter.
OUTPUTS	A list of the outputs produced.
EXTERNAL	A list of all external procedures invoked.
PROCEDURES	
AFFECTS	State alterations, etc. which are not explicit outputs, (e.g., globals effected, etc.). Modification of globals are discouraged.
HARDWARE	A list of hardware requirements and configurations necessary for proper

	medial function.
REAL-TIME REQUIREMENTS	List real-time execution requirements of the module.

1.5. SOURCE FILE MAINTENANCE

1.5.1 Module Storage. Program modules in the system shall be maintained in a file or data base by module type, or in a separate file or data base entities. There must be an external file name corresponding to the internal module(s) with additional information such as release and version. There must be an external file "qualifier" for the "type" of module in the file.

1.5.2 Module Creation and Updating. The source creation and maintenance shall be done in base levels. A base level is defined as a point at which the program source files have been frozen following successful testing. The base level will be maintained until major revisions and modifications justify the creation of a new base level. This point is usually reached whenever the overall function (sum of the individual functions) is significantly different from the previous release. This implies that more than a few revisions/modifications have been made for corrections and enhancements.

1.5.3 Post-delivery Maintenance. Once the equipment is installed at the designated Government site the contractor shall not maintain modem connections to the equipment or system, unless such connections are a specific requirement of the Statement of Work. Temporary modem connections desired by the contractor for remote system or applications software maintenance shall not be installed without the approval of the COTR. Such connections, if approved by the COTR, shall be installed in such a way that they are normally not connected, and the connection is made only through the deliberate and cognitive action of the Government staff. Such temporary connections (modems, switches, and cables) shall be clearly marked with red labels. Only under very rare circumstances will modem connections be allowed to equipment or systems once those systems are placed in operation by the Government.

1.6 SOURCE LANGUAGES

1.6.1 Languages. All machine instructions shall be generated using assemblers and/or compilers. High level languages (e.g., FORTRAN, COBOL, PASCAL) in wide use, will be preferred over assembly languages.

1.6.2 Compilers. Up to date (i.e., latest standard) compilers shall be used. When no industry standards are available, the source language must be in wide use on more than one vendors processor design.

1.6.3 Assemblers. Assemblers shall have macro capability.

2.0 SOFTWARE MAINTENANCE DOCUMENTATION

2.1 SCOPE

2.1.1 Scope. This document establishes a content standard to be observed by contractors in the preparation of software maintenance manuals. The manuals adhering to this standard shall be addressed to computer specialists (e.g., programmers).

2.1.2 Application. The requirements of this standard shall apply to systems developed or made to government specifications. Unless otherwise specified, they shall not apply to commercial production items which shall be supplied with commercial manuals.

The contractor shall provide all supporting documentation applicable to the delivered system, normally supplied (delivered) with commercial items by the Original Equipment Manufacture (OEM). In addition, the contractor shall provide a list of all the supporting documents available from the commercial OEM with an indication of applicability to the delivered system.

2.2 CONTENT

2.2.1 System/Subsystem Description. Provide a general description of the system/subsystem to establish a frame of reference for the rest of the document. Include a summary of the functional and data requirements satisfied by each system/subsystem. Describe the general interrelationships of the system/subsystem components. All external entities shall be identified and described. All descriptions must coincide with the diagrams mentioned in the following paragraph. List and describe limitations and operating constraints associated with the use of the system/subsystem. List and describe all hardware and software requirements of the system/subsystem, and modules.

2.2.2 System Diagrams. One or more system diagrams shall be included to show the complete configuration. These diagrams shall be structured from the general to the detailed, and be completely described by references on the diagram to textual descriptions. Subsystems and modules shall be related to the system. If the overall software system is complex, as determined by the COTR, each major subsystem shall be described in a separate section.

2.2.3 System/Subsystem Logic and Data Flow. Describe the logic and data flow of the entire system/subsystem in the form of a flowchart. The flow should provide an integrated presentation of the system/subsystem dynamics of entrances and exits, and computer programs.

2.2.4 Program/Data Module Descriptions. Complete documentation shall be provided for each program as described in the following subparagraphs.

2.2.4.1 Structure Charts. Hierarchal structure charts shall be drawn showing all modules, and the calling relationships between them.

2.2.4.2 Module Descriptions. Complete descriptions of each program module shall be included in the software documentation. The documentations shall include information as described in the following subsections.

2.2.4.2.1 Module Name, function, and calling sequence. List the name and function of the module and what it accomplishes. The name and function should serve as the index to the module in all related charts and cross references (e.g., source files). The name should be derived from the function, and conventions should be established, such that like naming occurs throughout. Secondly, the initial character or characters of names should be used in the message codes (e.g., following paragraph or messages).

2.2.4.2.2 Messages and Codes. All error messages and codes shall be listed in alphabetical order for use by the system operators. There must be complete explanations of what conditions generated the message. All error message codes should contain characters identifying the subsystem generating the message. A single document section describing the error messages shall contain the error codes and explanations for each subsystem. Possible operator actions in response to each message should be included in the message listings and/or in the supporting documentation.

2.2.4.2.3 Data Structure Description. Descriptions shall be given for each compound data structure used. Detailed descriptions shall be given for recursive structures.

2.2.4.2.4 Program Module Inputs. All inputs to a module shall be described in detail, including descriptions of the input format, and the internal storage format. This list of inputs must include those data items passed as arguments.

2.2.4.2.5 Program Outputs. Describe the output of the program and provide layouts (i.e., screen, printer, etc.).

2.2.4.2.6 Program Interfaces. Describe the interfaces with other software, such as data formats, messages, parameters, conversion requirements, interface procedures, and media.
2.2.4.2.7 Tables. Describe data tables used by one or more programs. The format, physical location (can be relative) and content must be completely clear with respect to each item within a table. Exceptions for security tables shall be approved by the COTR.

Tables shall be included in listing the total content of all ROM's. Procedures for updating tables shall be included in the User Documentation.

Symbol tables and cross reference listing shall be provided with each assembly language program.

2.2.4.2.8 Files. The access and use of data obtained from both temporary and permanent files shall be described. Refer to data base description requirements for complete documentation guidelines applicable to data maintained on peripheral devices.

2.2.4.2.9 Structured Psuedocode. An algorithmic description shall be given for each module describing the procedure undertaken by that module. This description shall be written in psuedocode, highly correlated with the actual code.

2.2.5 Operating Environment. The documentation shall include complete descriptions of the hardware and software used to support the system, as follows.

2.2.5.1 Hardware. Identify the equipment required for the operation of the system. Relate the hardware to each program. Include information such as:

a. Processor and size of internal storage;
b. Storage online or off-line, media, form, and devices;
c. Input/output devices, online and off-line;
d. Data transmission devices;
e. System memory map;
f. Interrupt and DMA structure, including priorities and defaults.

2.2.5.2 Support Software. Identify the support software needed for all computer programs (e.g., compilers, linkers, libraries).

2.2.5.2.1 Operating System. Identify and describe the operating system, including sysgen and input/output (I/O) generation (gen). Provide procedures, listings, and manuals.

2.2.5.2.2 Compiler/Assembler. Identify and describe the compiler or assembler including the version or release number and any special features used.

2.2.5.2.3 Other Software. Identify and describe any other software used.

2.2.5.3 Data Base. Describe or reference documentation provided for data base management. Include such information as codes, units of measurement, format, range of values, or reference a data element dictionary within which this information is contained.

2.2.6 Maintenance Procedures. A complete description of software maintenance procedures shall be provided with each system/subsystem (e.g., source generation, source edit, assembly/compiling, linking, sysgen (regeneration), initializing, etc.). Load lines shall be provided.

2.2.6.1 Programming Conventions. Identify and describe any programming conventions used. List these conventions along with the appropriate language or languages to which they apply.

2.2.6.2 Source Library Maintenance Procedures. Describe all file maintenance and update procedures used for source code. Include special products, techniques, and configuration management tools used.

2.2.6.3 Test and Verification Procedures. A complete set of procedures and data (or simulation systems) are to be described. Test and operational initialization procedures shall be included such that complete and partial replacement procedures for executable code are given.

3.0 SOFTWARE SYSTEMS USER DOCUMENTATION

3.1 SCOPE

3.1.1 Scope. This document establishes a standard for content to be observed by contractors in the preparation of User Manuals for software systems. Users for whom the manuals adhering to this standard shall be addressed are electronics technicians, console or terminal operators, and various applications specialists. The particular use to which each manual will be put depends upon what type of software is being used (e.g. technicians use of diagnostic software).

3.1.2 Application. The requirements of this standard shall apply to systems developed or made to Government specifications. Unless otherwise specified, they shall not apply to commerical production items, which shall be supplied with commerical manuals.

3.2 APPLICABLE DOCUMENTS

The following documents, of the issue in effect on the date of invitation for bid or request for proposal, form a part of this specification to the extent specified herein.

FIPS-PUB-38 "Guidelines for Documentation of Computer Programs and Automated Data Systems"

Federal Information Processing Standards Publications are available from: The Suprintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

NOAA/NESDIS Standard No. S24.801 "Preparation of Operation and Maintenance Manuals"

NOAA standards are available from: U.S. Department of Commerce, National Oceanic and Atmospheric Administration, National Environmental Satellite, Data, and Information Service, OSD/3, Washington, D.C. 20233.

3.3 CONTENT

3.3.1 General. The manual shall describe the functions performed by the software such that the system user/operator can determine their applicability and when and how to use them.

3.3.2 Overview. The manual should begin with an overview of the organization of the manual with directions as to the use of its content. The organization shall include content which provides: (1) a description of all capabilities of the system; 2) a complete description of the physical system and data flow; and 3) a description of the operation of all aspects of the system, with sufficient detail so as to provide a reference for both normal and abnormal (diagnostic and recovery) operations.

3.3.3 Detailed Operational Descriptions. Complete descriptions of each of the following items shall be contained within the manual or manuals provided by the contractor.

(1) A list and description of all user commands, sequences, and procedures.

(2) Capabilities of each command.

(3) Performance capabilities, including quantitative information, on inputs, outputs, communications and internal processes.

(4) Execution procedures for initializing each command macro.

(5) Data base update procedures for each type of field. This documentation requirement extends to updates of data stored within the computer memory as well as on disk or tape.

(6) Input descriptions (including definitions and formats) for all user inputs.

(7) Examples of all types of inputs showing all options for each input command and parameter field.

(8) Descriptions (including definitions and formats) of all output from the system.

(9) Descriptions of all error messages, their meaning and operator recovery procedures for each.

(10) Terminal set-up and operation for each terminal controlled procedure.

The documentation described above should be organized by subsystem and function. Manuals shall follow the format outlined in FIPS-PUB-38, "Guidelines for Documentation of Computer Programs and Automated Data Systems." (A format routinely used by the contractor may be used if pre-approved and accepted by the COTR.) References may be made to other available documentation for details of items such as for hardware operation (refer to NOAA/NESDIS Standard S24.801 -- "Preparation of Operation and Maintenance Manuals"). Frequent use of the diagrams and tables is recommended for ease of use by the system operators. An alphabetized index must be provided for operator use in locating information on procedures, commands, options, etc.

3.4 QUALITY ASSURANCE PROVISIONS.

3.4.1 Manual Review. Reviews of the manual shall be held during its preparation to assure compliance with this standard and preparation of a document of maximum usefulness. Manual reviews will be conducted at the contractor's facility, or a Government site, as mutually agreed. Areas of the document not providing clear and/or complete information for government personnel utilization of the system shall be noted for revisions to be performed by the contractor.

3.4.2 Draft Manual. A draft of the manual shall be prepared and submitted to the COTR for review at least 90-days prior to production of the final manual. If a training program is part of the contract, draft copies of the manual shall be used as a text to support the instruction. The final draft shall incorporate the corrections derived from the training program (if applicable), and other items, as directed by the COTR.

3.4.3 Final Manual. The final manual shall be prepared after approval of a final draft. Delivery shall be: a) one reproducible original; b) copies as defined by the procurement specification; and c) a complete set of magnetic media discs of in the wordproccessing format of the contractor. (Note: WordPerfect Corporation - "WordPerfect V6.0," or MicroSoft - "Word" are prefered by NOAA/NESDIS.)