IRM-5234-01
W/CH 1-2

U.S. Marine Corps

PROGRAMMING STANDARD

PCN 186 523401 00

DEPARTMENT OF THE NAVY
HEADQUARTERS UNITED STATES MARINE CORPS
WASHINGTON D.C. 20380

5234/01
CCIS-45

3 JUN 1987

From: Commandant of the Marine Corps

Subj: INFORMATION RESOURCES MANAGEMENT (IRM) PROGRAMMING STANDARD

Ref: (a) MCO P5231.1
(b) MCO 5271.1
(c) MCO P5600.31

Encl: (1) IRM-5234-01

1. PURPOSE. To provide guidance on programming standards within the Marine Corps as required by reference (a).

2. AUTHORITY. The information promulgated in this publication is based upon policy and guidance contained in reference (b).

3. APPLICABILITY. The programming standards outlined by this publication are applicable to all contractors and Marine Corps personnel who write programs for USMC automated information systems. This standard is applicable to the Marine Corps Reserve.

4. DISTRIBUTION. This technical publication has been assigned Distribution as listed below. Appropriate activities will receive updated individual activity Table of Allowances for Publications. Requests for changes in allowance should be submitted in accordance with reference (c).

5. SCOPE

a. Compliance. Compliance with the provisions of this publication is mandatory.

b. Waivers. Waivers to the provisions of this publication can be granted only by CMC (CC).

6. RECOMMENDATIONS. Recommendations concerning the constraints of this technical publication should be forwarded to CMC (CCI) via the appropriate chain of command. All recommended changes will be reviewed upon receipt and implemented if appropriate.

5234/01
CCIS-45

Subj: INFORMATION RESOURCES MANAGEMENT (IRM) PROGRAMMING STANDARD

7. SPONSOR. The sponsor of the technical publication is CMC (CCI).

[illegible]
R. R. PORTER
Brigadier General, U.S. Marine Corps
Director, Command, Control, Communications and Computer (C4) Systems Division

DISTRIBUTION:

L1P	
L1Q	
L1R	
7000006	(40)
7000031	(10)
7000007	(2)
7000012	(2)
7000014	(2)
7000015	(2)
7000045	(1)
7000050	(10)
7000142	(1)
7000148	(2)
7000066	(2)
7000093	(2)
7000020	(1)
7000062	(2)
7000017	(1)
7000019	(1)
7000070	(1)
7000067	(1)
7000071	(20)
7000072	(10)

Copy to: 8145001

DEPARTMENT OF THE NAVY
HEADQUARTERS UNITED STATES MARINE CORPS
WASHINGTON, D.C. 20380

IN REPLY REFER TO
5234/01 Ch 1
CCIS
14 Jan 88

From: Commandant of the Marine Corps

Subj: INFORMATION RESOURCES MANAGEMENT (IRM) PROGRAMMING STANDARD

Encl: (1) New page inserts to IRM-5234-01

1. Purpose. To transmit new page inserts to the basic technical publication.

2. Action

a. Remove present page v, page 2-1, and page 2-5, and replace them with corresponding pages contained in the enclosure hereto.

b. Insert new Appendix H, pages H-1 to H-8.

c. Change the "Distribution:" of the basic promulgation letter to read the same as shown in the "Distribution:" section of this Change.

3. Summary of Change. This Change adds an appendix regarding Ada coding standards.

4. Change Notation. Significant changes contained in the revised pages of this Change are denoted by an arrow (➡) symbol.

5. Filing Instructions. This Change transmittal will be filed immediately following the signature page of the basic technical publication.

6. Certification. Reviewed and approved this date.

[illegible]
R.W. COBBLE
Colonel, U.S. Marine Corps
Deputy Director, Command, Control, Communications and Computers (C4) Systems Division

PCN 186 523401 01

Subj: INFORMATION RESOURDES MANAGEMENT (IRM) PROGRAMMING STANDARD

DISTRIBUTION:

L1P	
L1Q	
L1R	
7000006	(40)
7000031	(10)
7000007	(2)
7000012	(2)
7000014	(2)
7000015	(2)
7000045	(1)
7000050	(10)
7000142	(1)
7000148	(2)
7000066	(2)
7000093	(2)
7000020	(1)
7000062	(2)
7000017	(1)
7000019	(1)
7000070	(1)
7000067	(1)
7000071	(20)
7000072	(10)

Copy to: 8145001

DEPARTMENT OF THE NAVY
HEADQUARTERS UNITED STATES MARINE CORPS
WASHINGTON, D.C. 20380-0001

IN REPLY REFER TO

5234/01 Ch 2
CCIS-45
16 JUN 1988

From: Commandant of the Marine Corps

Subj: INFORMATION RESOURCES MANAGEMENT (IRM) PROGRAMMING STANDARD

Encl: (1) New page inserts to IRM-5234-01

1. Purpose. To transmit new page inserts to the basic technical publication of 3 June 87.

2. Action

a. Remove present Appendix C, pages C-1 to C-6, and replace it with Appendix C, pages C-1 to C-6, contained in the enclosure hereto.

3. Summary of Change. This Change updates the appendix regarding COBOL coding standards.

4. Change Notation. Significant changes contained in the revised pages of this Change are denoted by an arrow (➡) symbol.

5. Filing Instructions. This Change transmittal will be filed immediately following the last page of the Change 1 transmittal letter.

6. Certification. Reviewed and approved this date.

[illegible]
R. L. PHILLIPS
By direction

PCN 186 523401 02

Subj: INFORMATION RESOURCES MANAGEMENT (IRM) PROGRAMMING STANDARD

DISTRIBUTION:

L1P	
L1Q	
L1R	
7000006	(40)
7000031	(10)
7000007	(2)
7000012	(2)
7000014	(2)
7000015	(2)
7000045	(1)
7000050	(10)
7000142	(1)
7000148	(3)
7000066	(2)
7000093	(2)
7000020	(1)
7000062	(2)
7000017	(1)
7000019	(1)
7000070	(1)
7000067	(1)
7000071	(20)
7000072	(10)
7000051	(4)

Copy to: 8145001

UNITED STATES MARINE CORPS

Information Resources Management (IRM) Standards and Guidelines Program

PROGRAMMING STANDARD
IRM-5234-01

3 JUN 1987

ENCL (1)

RECORD OF CHANGES

Log completed, change action as indicated.

Change Number	Date of Change	Date Received	Date Entered	Signature of Person Entering Change

PUBLICATION TABLE OF CONTENTS

Chapter 1

GENERAL

Chapter 1

GENERAL

1.1. OBJECTIVES. The objective of this standard is to establish a uniform methodology for the development of project software irrespective of its origin. This document provides guidelines for the development of program code in accordance with Detailed Design Specifications of the System Development Methodology as outlined in reference (a).

1.2. SCOPE. The areas of program development addressed in this standard cover COBOL, NATURAL, FOCUS and LIBRARIAN. The issues concerning program maintenance and the audit trails for program trouble reports or work requests will be covered in the project Configuration Management Plan.

1.2.1. Purpose. Use of this standard will ensure readable and supportable application program listings before they are distributed as a finished product. In the overall life of a system, the readability of the source code is more important than the techniques used in its creation. Additional areas of concern in program development are:

a. Clarity, simplicity, and unity of coding techniques used in the system development.

b. Clarity of program syntax, allowing an easy method for program evaluation.

c. Ease of future maintenance.

d. Overall support of the programming environment.

e. Portability of programs from environment to environment.

f. Loosely coupled and tightly cohesive modules.

1.3. APPROACH. The primary method for assuring the quality of delivered application programs is to have the programs adhere to the Detailed Design Specification (DDS) IRM-5231-06. This standard recognizes the importance of a good detailed design and therefore, assumes that acceptable design model structure charts, design data dictionaries, and module specifications have been built before the code is written. Where required, references will be made to the General Design Specification (GDS) IRM-5231-05 and the Detailed Design Specification (DDS).

Chapter Table of Contents

Chapter 2

STANDARDS

Chapter 2

STANDARDS

2.1. GENERAL. Programming deliverables should be developed in accordance with the standards described in the following paragraphs. It is understood that exceptions exist and should be allowed. However, all exceptions must be approved in writing prior to coding by (a) PROJECT MANAGER for in house development, or (b) the CONTRACTING OFFICER for all external development.

2.2. ENVIRONMENT STANDARDS. IRM 5234-04 will dictate naming conventions for Job Control Language (JCL), PROGRAMS, SYSIN MEMBERS, SCREENS and etc. Where a standard does not exist, local standards will be used until new Marine Corps standards are developed and published.

2.3. SUPPORT SOFTWARE. The following paragraphs will describe support software products available to users and developers, and will provide a reference to their usage.

2.3.1. LIBRARIAN. LIBRARIAN is a software system that can be used to manage both source amd object programs. LIBRARIAN control statements are used to add, delete, or update modules on LIBRARIAN. These control statements and their formats are described in the LIBRARIAN User Reference Manual. There are restrictions on the use of LIBRARIAN based on the installation. Restrictions may be found in the local SOP, which is the primary reference document for LIBRARIAN used at each site.

2.3.2. ROSCOE. ROSCOE is an on-line software system that provides interactive programming capabilities through the IBM 3270 terminals. Data can be entered, manipulated, and updated while being monitored on the screen. ROSCOE maintains a library with user unique prefixes for storage of user data. ROSCOE also provides a syntax checker for JCL and COBOL and provides a means to submit and execute jobs. It also permits the user to query job status and/or output of the job submitted for execution. The following manuals are available for more information on ROSCOE:

a. ROSCOE Command Reference Manual - Contains a detailed description of the available ROSCOE commands.

b. ROSCOE Handbook - Provides summary information concerning the commands and facilities of ROSCOE.

2.3.3. TSO with ISPF. TSO is a single user teleprocessing monitor that, with ISPF, provides a vehicle for the use of FOCUS, DESIGNMANAGER, and INFO MANAGEMENT, DATAMANAGER, CONTROL MANAGER. It provides foreground compiling and interactive debugging for COBOL programming when used as a development tool.

2.3.4. IBM Utilities. OS/VS provides utility programs to assist in organizing and maintaining data. There are three types of utility programs and they are classed by their function as follows:

a. System Utility Programs. Used to maintain and manipulate systems and user data sets. Entire volume manipulation, copying, or restoring is provided. These programs may be executed as jobs or can be invoked as subroutines; for example, IEBCOPY.

b. Data Set Utility Programs. Used to reorganize, change, or compare data at the data set and/or record level. These utilities manipulate partitioned, sequential, or indexed sequential data sets provided as input to the programs. Data set utility programs can be executed as jobs or can be invoked as subroutines by a calling program; for example, IEHPROGM.

c. Independent Utility Programs. They operate outside of, and in support of, the operating system. They are controlled by utility control statements and cannot be invoked by a calling program.

Refer to the IBM OS/VS2 MVS Utilities Manual for additional information concerning utility programs.

2.3.5. CA-SORT. CA-SORT Sort/Merge is a utility package which runs as a normal utility program and may be initiated using standard job control statements. The use of CA-SORT Sort/Merge is encouraged for project applications vice standard IBM sort products. This software normally handles all peripheral I/O device operations. Refer to CA-SORT Reference Guide for usage standards.

2.3.6. IDCAMS. IDCAMS is a portion of the IBM access method service utilities. IDCAMS deals with Virtual Storage Access Method (VSAM) files, including catalog definition, listing, etc.

2.3.7. ABEND-AID. ABEND-AID is a diagnostic aid used in debugging programs and is presently installed at the CDPAs. Refer to the CDPA SOP for usage.

2.4. APPLICATION DEVELOPMENT STANDARDS.

2.4.1. COBOL. Appendix C describes the procedure that will be used in developing application programs using COBOL.

2.4.2. NATURAL. Appendix D describes the procedures for use of NATURAL.

2.4.3. FOCUS. Appendix E describes the procedures that will be used in developing application programs using FOCUS.

2.4.4. ADABAS. The goal of the standards established in this section is to achieve uniformity in the use of ADABAS. It is understood that this document cannot cover all programming situations. For those areas not covered, the Data Base Administrator is the appropriate authority.

a. General Program Design. Appendix F describes which standards apply to the general design of user written programs which interface to the ADABAS data base via ADAMINT. Appendix G describes the general design of user written programs which interface to ADABAS data base via direct calls.

2.4.5. VSAM. All data files not on ADABAS should be VSAM if stored on Direct Access Storage Device (DASD).

2.4.6. Ada. Appendix H describes the procedures that will be used in developing application programs using Ada.

2.5. EVALUATION CRITERIA. Delivered source code must display the following characteristics:

2.5.1. Standards. The code must not violate the standards listed in Appendixes C through G.

2.5.2. Source Code. The source code must be compared to the Detail Design Structure Chart on which it is based and which was built during detailed design.

a. For each module on the structure chart there must be one or more paragraphs of code that implement that module.

b. For each paragraph of source code, there must be a module on the structure chart of which the paragraph is a part.

c. If either of these conditions is not true, the code is not acceptable as delivered source code.

Appendix A

GLOSSARY

DAS PLAN: DAS is an acronym for "Data Access Security."

OS/VS: OS is an acronym for "Operating System." VS is an acronym for "Virtual Storage."

TSO with ISPF: TSO is an acronym for "Time Sharing Option." ISPF is an acronym for "Interactive System Productivity Facility."

Appendix B

REFERENCES

1. MCO 5271.1 - Gives detailed technical direction and guidance governing the internal management of Marine Corps ADP activities. It provides detailed operations of the MCCDPA's at the field level.

2. IRM-5231-06 - Detailed Design Specification.

3. LIBRARIAN User Reference Manual (SLG-10-59) -- Contains the LIBRARIAN control statements and their use.

4. CA-SORT Reference Guide (Release 6.1) -- Provides guidance in the use of SORT software.

5. Optimizer III User Guide (D1170C-47(5c)1071) -- Provides information on the optimizer for COBOL programs.

6. ROSCOE Command Reference Manual (SR20-30-20) -- Contains a detailed description of the available ROSCOE commands.

7. ROSCOE Handbook (SRQO-20-20) -- Provides summary information concerning the commands and facilities of ROSCOE.

8. American National Standard COBOL X3.23-1974 (ANS COBOL-74), (Version 3, Release 3.3) -- To be used as a reference manual in the writing of COBOL programs.

9. OS/VS2 MVS Utilities Manual (GC26-3902-0) -- Provides utility programs to assist in organizing and maintaining data.

10. An Introduction to Structured Programming in COBOL (GC20-1776-0) -- Describes and illustrates the use of structured programming.

11. Access method services (Release 3.8) GC26-3841-3 -- Contains all VSAM utilities, including IDCAMS.

12. ADAMINT Programmers Reference (ADA-471-054) -- Provides detailed descriptions of called ADABAS functions.

13. NATURAL Introduction Manual (NAT-120-001) -- Provides an introduction to NATURAL.

14. NATURAL Users Manual (NAT-120-020) -- Provides an in-depth introduction to NATURAL and its' use.

15. NATURAL Reference Manual (NAT-120-030) -- Describes the syntax, function, and use of the NATURAL statements,

parameters, variables, and functions.

16. Reference Manual for IBM 3350 Direct Access Storage (GA26-1-638) -- Contains detailed information on space allocations for the IBM 3380 disk drives.

17. CICS/VS Application Programmers Referance Manual Command Level (SC33-0077) -- Contains User level instructions for application programmers.

18. CICS/VS Messages and Code (SC33-0081-3) -- Contains error messages.

19. CICS/VS IBM 3270/8775 Guide (SC33-0157-0) -- Contains User information for application programmers relating to Terminals.

20. Focus Users Manual (Release 5.0) -- Contains user level information for application programmers/end users.

21. IRM-5234-04 - Naming Conventions.

22. IRM-5510-01 - Data Access Security.

Appendix C

COBOL

1. All COBOL programs will be developed in accordance with American National Standard COBOL, X3.23 - 1974 (ANSI COBOL-74), Version 3, Release 3.3. This appendix provides the basis for VS COBOL II standards which will be published at a later date. The following conventions will also be employed:

a. All table searches will be performed using the SEARCH or SEARCH ALL verb. Programmer written search routines may be used in special cases. Predefined members are available and are referenced in the site SOP. HIGH-VALUES or LOW-VALUES will be used to fill a partially loaded table.

b. To report edit errors and exceptions, all programs will either produce an error listing or send records to an output report dataset. Programs may contain programmer designed ABENDS if sufficiently warranted and documented in operators manuals and users manuals.

c. Programs will not require input from nor produce output to the operations console. The only authorized exception to this is payroll check processing.

d. CA-SORT is the preferred sort utility. The USING file-name GIVING file-name constructs of the COBOL sort feature will not be used. The COBOL sort feature will only be used with the input-procedure and/or output-procedure constructs and only when a limited volume of records will be handled.

e. Use of the COBOL report writer is prohibited. Standard report printing logic and formats will be used whenever possible.

f. Output should be printed eight lines per inch on 8½ x 14 7/8 inch paper unless other spacing and forms are required by the application.

2. All application program source code (e.g., file descriptions, record descriptions, internal tables, and reusable program codes) will reside on a LIBRARIAN disk master file. It will be called by programs through the LIBRARIAN INCLUDE feature, or the COBOL COPY verb as it applies to LIBRARIAN. Source code will not be accepted for inclusion from partitioned data sets with the exception of operating system routines. For additional information concerning the use of LIBRARIAN, see Appendix B, "References".

3. COBOL programs will consist of four divisions: INDENTIFICATION, ENVIRONMENT, DATA, and PROCEDURE.

a. <u>IDENTIFICATION DIVISION</u>

PROGRAM-ID	Eight character program name.
AUTHOR	Author's name, team name.
DATE-WRITTEN	Date of original program coding.
DATE-COMPILED	Supplied by the compiler.
INSTALLATION	Name of responsible ADP acitivity.
SECURITY	Highest security classification contained in the source code.
* IN COLUMN 7	Brief statement of program's or module's function. Reiteration of comments from Detailed Design Specification or Mini-Specification.

b. <u>ENVIRONMENT DIVISION</u>

CONFIGURATION SECTION (Optional)

SOURCE-COMPUTER	Hardware system for compilation
OBJECT-COMPUTER	Hardware system for execution

c. <u>DATA DIVISION</u>. The DATA DIVISION will contain a WORKING-STORAGE SECTION and a LINKAGE SECTION. All record descriptions will be placed in the WORKING-STORAGE SECTION.

(1) <u>WORKING-STORAGE SECTION</u>. The following guidelines will be followed when coding the WORKING-STORAGE SECTION:

(a) The first and last entries of the WORKING-STORAGE SECTION are:

```
01   FILLER                          PIC X(44)
     VALUE "PROGRAM XXXXXXXX WORKING-STORAGE BEGINS HERE".

01   FILLER                          PIC X(42)
     VALUE "PROGRAM XXXXXXXX WORKING-STORAGE ENDS HERE".
```

(b) Each record or 01 description will be preceded by two blank lines so that groupings are visibly separated.

(c) All 01 levels will be in alphanumeric order.

(d) Level numbers will be incremented by 2 or more (for example: 01, 03, 05).

(e) Level-88 conditions are to be indented four spaces from the variable to which they refer.

(f) No level-77 entries will be used.

(g) Variable names will be on the same line as the level number and separated from the name by two spaces.

(h) Each successively lower level will be indented four spaces from the previous higher level.

(i) All PICTURE (PIC) clauses will begin in column 36. If a data name extends past column 34, the PIC clause will be placed in column 36 of the next line. The number of characters in a PIC clause will always be indicated by a number (for example: 03 DATA-NAME PIC X(2)).

(j) Numeric data used for computation will be defined as COMP-3.

(k) DISPLAY data fields not referenced by arithmetic statements will be described as "X" rather than "9".

(l) VALUE clauses start in column 48. If the VALUE clause does not fit in the remaining space, the entire clause will be written on the next line starting in column 16.

(m) VALUE clauses assigning numeric values to a data field defined as alphanumeric must enclose the numeric literal in single quotes.

(n) When more than one line is needed for a WORKING-STORAGE entry, the continuation lines will be indented four spaces.

(o) The data names used in COBOL programs will be the same as stated from the Detail Design Data Dictionary derived from the Detail Design Specification during the detail design phase of the system life cycle.

(p) Index names, names of temporary fields, and other data items not in the Detail Design Data Dictionary will have names that describe their function.

(q) All variables used as subscripts will be defined as COMP.

(r) Each data name used as a switch or subscript will be used for only one purpose and not reused for a second function.

(2) LINKAGE SECTION. Data passed through the LINKAGE SECTION will be defined under a single 01 record if possible. This record is included in both the calling and the called program.

d. PROCEDURE DIVISION. The following conventions apply to the PROCEDURE DIVISION of the program:

(1) For every module on the Detail Design Structure Chart, there will be one identifiable block of code. Modules are implemented in one of three ways:

(a) Modules are implemented as SECTIONS. The section ends with an exit paragraph.

(b) Modules are implemented by a series of paragraphs ending with an exit paragraph.

(c) Modules are coded in-line with comments marking the beginning and ending of the module.

(2) Each module will be preceded by two blocks of comments.

(a) The first block of comments will contain the following information, if applicable.

1 Module Name, full Mini-Spec Name.

2 "Including" Module Name(s) and full Mini-Spec Name.

3 "Including" Module Name(s) and full Mini-Spec Name and any Called Modules and function names.

(b) The second block of comments will be a reiteration of the relevant comments contained in the Mini-Spec from which the module is derived.

(3) A third block of comments will terminate each module with 'END MEMBER member-name full Mini-Spec name'.

(4) Literals 1 and 0 used to initialize indexes, subscripts, and accumulators will be the only literals in the PROCEDURE DIVISION. All other constant data values are established in the WORKING-STORAGE SECTION as named constants.

(5) Reserved words GREATER THAN, LESS THAN, and EQUAL TO will be used in conditional statements rather than the symbols >, <, and =.

(6) ALTER, APPLYCORE, EXAMINE, NOTE, POSITIONING, INDEX, and MOVE CORRESPONDING will not be used in any COBOL programs.

(7) Each section name and paragraph name will begin in column 8. Each section will be preceded by two blank lines. Each section and paragraph name will be preceded by a four digit number to allow easier reference.

(8) Sections and paragraphs will be placed in numeric sequence within the PROCEDURE DIVISION. Control section/paragraph will begin and end with:

```
0000-BEGIN-CONTROL-(SECTION/PARAGRAPH)

0000-END-CONTROL-  (SECTION/PARAGRAPH)
```

Each section/paragraph will be numbered in increments of 1000. Each subparagraph performed will be in increments of 100, followed by subparagraphs within these paragraphs incremented by 10, followed by 1's. Thus paragraphs 1000, 1100, 1110, 1111 are all related.

(9) As a rule, a paragraph will not exceed one page in length and will normally correspond to a design module. However, paragraphs which are straight forward in nature with a low level of complexity may exceed one page in length. In these cases, dividing a paragraph into two paragraphs may actually make it more complex and difficult to follow.

(10) The paragraph name will be taken from the module name on the Detail Design Structure Chart.

(11) A statement which begins a COBOL sentence will start in column 12. There will be only one statement per line. If a statement requires more than one line, successive lines are to be indented four spaces.

(12) All reads and writes will use the READ INTO, WRITE FROM constructs so that all manipulation of data records is done in the WORKING-STORAGE SECTION. Each statement option of the READ and WRITE verbs will be coded on a separate line.

(13) GO TO will not be used except for GO TO DEPENDING ON and GO TO paragraph-EXIT.

(14) All called programs will use GO BACK rather than STOP RUN.

(15) When using the IF/THEN/ELSE statement, the ELSE is placed on its own line and will begin in the same column as the corresponding IF. Statements following the condition and the ELSE will be indented four spaces.

(16) For a series of Mutually Exclusive IF statements, all IF statements will begin in the same column. There is no limit to the number of IF statements when all conditions are mutually exclusive.

(17) For Nested IF statements, each level of the IF will be indented four spaces. No nested IF structure will contain more than 3 levels. Up to 5 levels may be authorized in writing by the project manager if deemed necessary. For example:

```
       Nested IF                 Mutually Exclusive IF

    IF condition-p                 IF condition-p
        statement-1                    statement-1
        IF condition-q             ELSE
            statement-2            IF condition-q
            IF condition-r             statement-2
                statement-3        ELSE
            ELSE                   IF condition-r
                statement-4            statement-3
        ELSE                       ELSE
            statement-5            IF condition-s
    ELSE                               statement-4.
        statement-6.
```

(18) All PERFORM THRU statements will specify an exit paragraph name as the terminating paragraph.

(19) COMP-1 and COMP-2 will not be used.

(20) Nested OCCURS DEPENDING ON form will not be used. This eliminates variable length entries in external tables (i.e., all table entries must be of fixed length).

4. <u>LINKAGE EDITOR</u>. All executions of LINKAGE EDITOR for COBOL programs will employ the SETSSI feature. SETSSI statements specify hexadecimal information to be placed in the system status index of the directory entry for the output module. This feature is employed to provide a uniform DATE-TIME stamp for all compilation-link edits. SETSSI will be used on end products and exported products.

Appendix D

NATURAL

NATURAL Conventions

NATURAL conventions are contained in the following documentation manuals:

NATURAL Introduction Manual (NAT-120-001) - This manual provides an introduction to NATURAL.

NATURAL User Manual (NAT-120-020) - This manual provides an introduction to NATURAL at the user level giving a detail explanation of the functions and commands for NATURAL.

NATURAL Reference Manual (NAT-120-030) - This manual describes the syntax, functions, and use of the NATURAL statements, parameters, variables, and functions.

Specific conventions are outlined in the sections that follow:

Program Identification

The following conventions apply to NATURAL program identification:

a. Programs will be identified by an eight position name.

b. ACCESSOR ID, which is specified in the LOGON command, will be designated by the CDPA Security Office which are specified then by the LOGON command.

   -- Private libraries are not allowed. Special cases will be considered and created by the authority of the Data Base Administrator (DBA). Unauthorized libraries and their contents will be deleted if found.

   -- A System Library is used for storage of programs. This library is controlled by the DBA at each CDPA.

   -- There is an identified Program Manager assigned for each library.

c. Program names are assigned by the Program Manager, and the program names will be indicative of their functions.

d. The internal description is a set of comments that is required in each stored program. These comments must include the IDENTIFICATION and HISTORY SECTION.

Identification Section

The following conventions apply to the IDENTIFICATION SECTION of a NATURAL Program.

a. Program-ID - Program names will consist of eight characters. Positions 1 through 5 will be the same as the job name, and positions 6 through 8 will be assigned by the Program Manager.

b. Author - The author and section names are required. If a team is responsible, then the author may be the chief programmer.

c. Installation - The entry is to be defined at each CDPA.

d. Date-Written - This field will contain the completion date of the program.

e. Files Accessed - All files used by a program will indicate read only or update as approriate.

f. Programs Called - All programs which are called by this program will be listed.

g. Called By - All programs which call this program will be listed.

h. Maps used - Names of all maps used by the program.

i. Mode of operation - On-line or batch.

j. Functional Manager - Name of position or office responsible for program modifications.

k. Security - The highest security classification level including "Unclassified" that this program contains will be stated.

l. Remarks - The remarks section will contain at least the following information:

   (1) What the program does.

   (2) Points of contact - No names; just section/billet.

   (3) Phone numbers for the points of contact, the primary user of the program, and the development section that produced the program.

History Section

The following minimum information should be contained in the HISTORY SECTION:

a. Modification Date - This area will identify all changes made to this program.

b. Modification Description - This area will contain a description (relating back to a date of modification) of how the modification will affect the program.

c. Modified By - This area will contain the name and phone number of the person or section making the modification.

Program Support Section

The following standards and conventions should be used in coding the PROGRAM SUPPORT SECTION:

a. Set Globals - This statement will always be coded as SG=OFF if negative values are not to be encountered. If negative values are to be checked for then SG=ON must be used.

b. Reset Variables - All variables will be set prior to the execution of the program.

Main Code Section

The following conventions should be used in coding of the MAIN CODE SECTION:

a. Only one statement is allowed per line of NATURAL code.

b. NATURAL DDN file names will be used instead of ADABAS file numbers.

c. NATURAL field names will be used instead of the ADABAS two character field names.

d. ALL processing loops will be closed with the CLOSE LOOP statement.

e. Variable field names are to begin with the # character and are to be defined at the beginning of the program or subroutine with the RESET or the ASSIGN statement unless the SORT BY command is used.

f. Numeric fields input from a terminal will be displayed with SG=OFF unless a negative number is expected.

g. The following conventions are to be used for identation in coding a NATURAL program:

-- Verbs which are to be conditionally executed (IF), will begin on a separate line, indented by four spaces.

-- DO and DO END statements will be indented by four spaces from the condition.

-- ELSE statements will start in the same position as the condition or on the next line.

-- Subsequent nested conditions will be indented by four spaces.

-- Statements within a processing loop will be indented by four spaces from the loop initiating statement.

Performance

The following tips should be considered to prevent system/program degradation during execution:

a. FIND NUMBER should not be followed by a FIND with the same search criteria.

b. HISTOGRAM should be used when all that is required is a count of a descriptor or as validation.

c. Replace the FIND WITH using two descriptors by a FIND on the most restricting descriptor and WHERE on the second descriptor.

d. Use a FIND and RETAIN the ISN list when the same records are used more than once.

e. Use the READ PHYSICAL and external SORT vice READ LOGICAL when at least 30% of a file is processing in a batch mode.

f. Use INPUT NO ERASE when redisplaying most of a screen.

g. Use REPEAT/LOOP statements in any iterative programs vice having a program "FETCH" itself.

h. Packed format should be used for numeric fields.

i. OBTAIN should only be used to return the occurrences that are guaranteed to be existing.

j. Avoid excessive ET's where possible.

k. In update program logic, if the program is not a one-for-one READ AND UPDATE, a GET BY ISN must precede the UPDATE command to prevent placing rejected records "on hold" and impacting other data base users.

Appendix E

FOCUS

FOCUS standards currently under development by the SDM work group.

Appendix F

ADAMINTS

a. All programs will be designed and constructed to properly function as a user within the multi-user environment of ADABAS. This further implies that the system builder is aware that each program is operating in a shared-resource environment with upper limitations and that at any instance in time the program could have significant impact upon other users or the successful performance of the shared resources.

b. No program will require exclusive control of an ADABAS file unless specific system specifications require its use. Therefore, on-line update programs will be designed in a manner to notify the terminal user that the specific record currently attempting to be updated is being "HELD FOR UPDATE" by another user. Batch programs should normally include retry logic to obtain exclusive control at the record level prior to program abort if the situation warrants. To accommodate this logic constructing the appropriate ADAMINT function (e.g., FINDSET, etc.) will utilize the HOLD=RETURN option to provide immediate ADABAS return code, signifying the "RECORD HELD" condition without invoking ADABAS automatic retry logic, to the user program.

c. All programs which update the data base will incorporate END TRANSACTION (ET) logic to indicate that the user's current logical transaction is complete. In an on-line application all update activity caused by one update screen would be considered as one logical transaction. Therefore, an ET should be issued for each update screen. User defined restart data is optional dependent upon circumstance of restart requirements. The program must issue the ET or other such command within the global constraints of the transaction time limit (TT) to preclude inadvertent "time out" conditions.

d. Although no specific number of records which may be held for update can be mandated, the user can draw certain conclusions from the ensuing items.

I. It would be ill advised for an on-line program to capture several transactions from the terminal and suddenly require the user to re-input the same information because the program was "timed out" or the shared resource failed due to hardware or software reasons.

II. Other concurrent users might have need to obtain a record for update which is being held even though all updates have been applied.

III. The sum of all records to be held for update by all users at any one given time cannot exceed a predetermined number set at session initialization time of ADABAS. The HOLD QUEUE SIZE (NH) value is established by the DBA/SDBA and may be adjusted upward as need arises. ADABAS itself provides no control mechanism nor are there any user written interface to preclude any one user from completely obligating the entire record hold queue and serverly implicating all other users.

e. All programs which update the data base will be restartable. Batch programs will issue non-synchronized checkpoints. These checkpoints will be issued twice during the execution of a program; once when program is entered for execution and once prior to successful program termination. Each checkpoint is to be identified so as to be able to distinguish the difference between the beginning and ending checkpoint. The express purpose of this convention is to facilitate the use of ADABAS recovery functions (regenerate or backout) when required.

f. Any batch program which contains extensive NON-ADABAS related restart logic that might result in a lengthy time delay, between the initial SIGNON function and a subsequent ADABAS command, should take appropriate measures to prevent an inadvertent "time out" condition that might arise due to ADABAS non-activity parameters.

g. All programs will issue calls to the ADABAS functions SIGNON/SIGNOFF. Due to performance considerations, it will be at the discretion of the DBA/SDBA to determine if the ADAMINT function will actually perform the command by the option CALLADA parameter setting.

h. If ADABAS file(s) password(s) are to be utilized as part of the overall security scheme all on-line functions will include some means of obtaining user input so as to ensure that the feature is dynamic and can be rapidly changed, if compromised, without program recompilation. If the system contains batch programs which access secured ADABAS files then the ADAMINT used by each program will contain the appropriate ADABAS file(s) password(s).

i. To the degree possible, all on-line programs will be designed in such a manner to achieve the highest possible degree of data independence and change impact. In this regard, system builders are reminded that task management functions (COLOADS, COLINK, etc.) can be equally applied to ADAMINTS as well as programs.

j. Certain ADABAS functions are resource intensive and should be used only when all other solutions to a given processing requirement have been thoroughly evaluated. A FINDSET operation that uses the ADABAS SORT feature is a typical example of this and, as a general rule, should be avoided.

k. ADABAS conveniently supplies the user program the mechanics to conform to top-down program design where after a set of records have been selected via the FINDSET operation, it can additionally "read the first data record" leaving the "read next record" to the READSET operation. Therefore, all programs regardless of expected record quantity will anticipate data being returned upon a call to a FINDSET operation. This requirement has the desired effect on performance in that even if the FINDSET selection criteria happens to be for a descriptor with unique properties then the function is completed by the Sx vice Sx Lx command sequence.

l. In most cases there is more than one solution to the manner by which a program will cause itself to navigate through the data base in order to arrive at the minimum data which satisfies a data request. Each solution will have its own individual merit, however; the acceptable solution will be that which is most effective and efficient from the data base perspective. Although this standard might appear to be vague, the reader should understand that due to the number of variables involved it could not begin to discuss the merits of any given general approach. The intent, therefore, leaves no doubt that it is unacceptable to select one solution over another based upon its "ease of use", if such solution causes considerable more drain on total system resources when all elements of the data base are considered.

m. Batch programs can achieve a higher degree of insulation against physical data base change such as a file number change than can on-line programs. Therefore, it is strongly recommended that all batch programs contain CHGFILE logic. If similar logic can be incorporated in the on-line environment it is considered as favorable. Logic of this nature accommodates those situations where the same userview (ADAMINT) operates against the same logical file structure when both the test and production versions of the target file(s) reside in the same physical data base.

Appendix G

DIRECT CALLS

Direct calls standards currently under development by the SDM working group.

Appendix H

ADA

1. Purpose. The purpose of this appendix is to specify default coding standards for programs written in the Ada* Programming Language. The standards are designed to increase the readability, reliability, and portability of Ada programs.

2. Applicability. This appendix applies to all Ada source code.

3. Ada Programming Language. Subject to other provisions in this appendix, the Ada Programming Language standard ANSI/MIL-STD-1815 shall be used:

a. To define valid Ada code.

b. To construe Ada words and phrases.

c. To provide Ada code examples.

4. Ada compilers. If one or more Ada compilers are specified, then all code shall be compilable by the specified Ada compiler(s).

5. Detailed requirements

5.1 Introduction

5.1.1 Ada prolog. In order to facilitate automated source code documentation (e.g., cataloging, indexing, searching, retrieval), a standard prolog for Ada source code shall be provided with every Ada compilation unit and with every Ada subprogram, package, or generic unit that exceeds 200 lines of code.

5.1.1.1 Every Ada prolog shall consist of three blocks of Ada comments. These blocks shall provide information about the unit as a whole as well as about every enclosed Ada subprogram, package, and generic unit that does not contain its own Ada prolog. The Ada prolog blocks shall be consecutive and shall be formatted as follows:

a. An abstract block that summarizes or outlines the purpose and function of the unit(s).

b. A block of keywords that capture the purpose and function of the unit(s) with index words and phrases.

* Ada is a registered trademark of the U. S. Government/Ada Joint Program Office

c. A block that describes technical contents and administrative details of the unit(s).

5.1.1.2 The abstract block shall be a contiguous block of not more than 10 lines containing only Ada comments, followed by a blank line. The first line of the block shall contain the 8-character string "ABSTRACT" followed by unit identification strings such as unit name, search key, or library code. The rest of the block shall contain the abstract.

5.1.1.3 The block of keywords shall be a contiguous block of not more than 10 lines containing only Ada comments, followed by a blank line. The first line of the block shall contain the 8-character string "KEYWORDS" followed by the same unit identification strings used with the abstract block. The rest of the block shall contain the keywords and phrases, separated by commas.

5.1.1.4 The contents shall be a contiguous block of lines containing only Ada comments, followed by a blank line. The first line of the block shall contain the 8-character string "CONTENTS" followed by the same unit identification strings used with the abstract block. The rest of the block shall include the following information for the unit(s):

a. Creation information: date, organization, author.

b. Revision history: For every revision: date, change summary, authorizing document (e.g., problem report, change request), change organization, section, author.

c. Unit purpose: What the unit does.

d. Unit function: How the unit achieves its purpose (e.g., algorithms, structures, objects, types, tasks).

e. External Ada units accessed (e.g., called, imported, WITHed, referenced): Unit name, access purpose.

f. Description of every Ada exception potentially propagated by this unit.

g. Input/output to all devices (e.g., files, keyboards, consoles, printers, modems, sensors, ports): Device name, access type, access purpose, access summary.

h. Machine-dependencies (e.g., registers, memory, bulk storage, ports, machine code): Access type, purpose, and justification.

i. Compiler-dependencies (e.g., special pragmas, optional implementation, specific compilers): Special requirements and justification.

5.2 Lexical elements

5.2.1 Explanatory comments. Source code shall be supplemented with Ada comments that explain the code. Explanatory comments shall not duplicate the Ada syntax or semantics, but shall clarify the encoded data structures or process algorithms at a more descriptive level than the code. Comments shall be grammatically and technically correct and shall address a reader who is an Ada programmer.

5.2.2 Command comments. When automated code manipulation utilities are used that employ specially formatted Ada comments as commands, such command comments shall be distinguished and isolated from explanatory comments.

5.3 Declarations and types

5.3.1 Types and subtypes. Types and subtypes shall be used:

a. To group objects with similar properties or operations.

b. To constrain properties of objects.

c. To support error detection at compile and run times.

5.3.1.1 Objects that represent different domains, structures, or sets shall be declared as different types.

5.3.1.2 Objects of the same base type whose value sets are used with different ranges or constraints shall be declared as different subtypes.

5.3.2 Initialization of variables. All variables shall be assigned an initial value by elaboration, assignment, or indirect updating prior to their use. No assumption shall be made about the value of an uninitialized variable.

5.3.3 Real numbers. Real numeric variables shall be declared using the RANGE, DIGITS, and DELTA clauses.

5.3.4 Constants and named numbers. An object whose value will not change after each elaboration shall be declared as a constant. A numeric object whose value will not change after compilation shall be declared as a named number.

5.3.5 Derived constants. Constants and named numbers derived from named objects shall be declared using the object names rather than literals (e.g., declare PI/2 rather than 1.5708).

5.3.6 Literals. The use of literals shall be permitted only for working constants. Names shall be declared for all values mentioned in the program specifications.

5.4 Names and expressions

5.4.1 Name qualification. The use of all Ada name forms and qualification features shall be permitted.

5.4.2 Naming conventions. Any naming convention shall be permitted, provided it:

a. Uses names that describe and distinguish entities.

b. Uses names whose English syntax reflects the nature of role of entities (e.g., verbs for actions, nouns for data, adjectives for status).

c. Uses abbreviations, acronyms, and mnemonics only where justified by context or common usage.

d. Enhances program readability.

e. Is consistently applied throughout a compilation unit.

f. The first 8 characters of any file or program name must be unique.

5.4.3 Expressions. The order of evaluation for compound expressions shall be clarified with parentheses and spacing.

5.4.3.1 Expressions shall not be nested more than four levels below the Ada statement level.

5.4.3.2 Expressions with more than one negated logical operator shall be prohibited. Membership tests and short-circuit control forms shall be used where applicable.

5.4.3.3 Mixed mode operations that are not explicitly declared or specifically supported by typed operations shall be accompanied by Ada comments.

5.5 Statements

5.5.1 Source code text. Source code text shall emphasize logical structure and enhance readability, using the following style:

a. Each line shall contain at most one statement.

b. Related code such as declarations, loops, blocks, cases, and exception handlers shall be grouped, separated with blank lines, and described with Ada comments.

c. Subordinate code such as nested statements, continuation lines, and embedded units shall be indented and aligned.

d. When there is a choice between alternate constructs that meet all applicable specifications and standards, the more readable construct shall be used.

e. The same presentation, nesting, and indentation style shall be used consistently throughout a compilation unit.

5.5.2 Loop termination. The conditions that terminate a control loop shall be identified with Ada comments or statements such as FOR, WHILE, EXIT, RETURN, or RAISE. Non-terminating loops shall be permitted only where required by program specifications, and shall be identified with Ada comments.

5.5.3 GOTO statement. GOTO statements shall be used only where required to meet specified execution time or space constraints. Every GOTO statement so used shall be accompanied by:

a. Comments placed near the GOTO statement to document the applicable constraints.

b. Comments placed near the statement receiving control.

5.6 Subprograms

5.6.1 Use of subprograms. The use of Ada subprogram units shall be permitted:

a. To modularize data and processes into decoupled and parameterized procedures, functions, and operations.

b. To structure the control flow and make it readable.

c. To constitute the library modules shared by many units.

5.6.2 Subprogram coupling. Subprograms that are not embedded in the same package shall share data, logic, code, or control only by passing formal parameters at invocation time. Exceptions to this rule shall be permitted where required to meet program specifications. In such cases, explanatory Ada comments shall be included in each participating subprogram.

5.7 Packages

5.7.1 Use of packages. The use of Ada packages shall be permitted:

a. To structure data and processes into abstract types, objects, algorithms, tasks, state machines, or domains.

b. To organize the Ada program and provide information about it.

c. To group related Ada units such as declarations, operator pools, cooperating tasks, closely coupled subprograms, and subprogram libraries.

d. To isolate Ada units with special information hiding requirements such as classified procedures, machine or device dependent code, patented algorithms, and dynamically generated or altered code.

5.7.2 Private types. Private and limited private types shall be used to encapsulate abstractions and to hide implementation details at execution time. The private part of a package shall not contain more code than required to define the private type.

5.7.3 Imported code. The source code for all Ada units that are directly or indirectly imported, called, WITHed, or otherwise referenced by a package shall be deemed a part of that package and shall meet all applicable specifications and standards.

5.8 Visibility

5.8.1 Overloading. Operator overloading shall be permitted, provided that the same operator symbol shall not be used for contradictory operations (e.g., use "+" for accumulating or expanding, not for depleting or negating).

5.9 Tasks

5.9.1 Use of tasks. The use of Ada tasks shall be permitted:

a. To implement concurrent processing and interprocess communication.

b. To express algorithms requiring parallel processors or mulitprocessing.

c. To clarify, abstract, or simplify other algorithms not necessarily involving concurrency.

5.9.2 Task scheduling. No assumptions shall be made about the relative order or execution speed of different tasks, except those implied by Ada rendezvous semantics. The pragma PRIORITY shall be relied on for task scheduling only if a supporting multiprocessing system with task scheduling is specified as part of the program.

5.9.3 ACCEPT statement. Within an ACCEPT statement, only those actions shall be performed that must be completed before control can be returned to the calling task.

5.10 Program structure and compilation units

5.10.1 Separate compilation units. Provided that each Ada unit as well as the entire Ada program shall meet all applicable specifications and standards, the use of separate Ada compilation units shall be permitted:

a. To organize the program into a physically decoupled, less complex, and more readable structure.

b. To support automated preparation and maintenance of required program documentation.

c. To reduce recompilation required to make and test changes.

d. To promote the reusability of Ada units and the use of Ada unit libraries.

5.10.2 WITH clause. Whenever possible, the WITH clause shall be included in the body rather than in the specification part of an Ada unit. If only the body is dependent, the context clause shall not be included in the specification part.

5.10.3 USE clause. The USE clause will not be used.

5.11 Exceptions

5.11.1 Use of Ada exceptions. Provided that Ada exception code shall be subject to the same permissions and prohibitions as all other Ada code, the use of Ada exception features shall be permitted:

a. To detect or signal erroneous, anomalous, or sporadic events, conditions, data, or machine states.

b. To handle exceptional events or situations.

c. To separate code for normal and exceptional control flow.

5.11.2 Exception checking. Built-in checking shall not be suppressed except where required to meet specified execution time or space constraints. In such cases, the SUPPRESS pragma shall be used with the minimum scope required, and only on algorithms which can be shown to conform to the unchecked constraints.

5.11.3 Exception handling. When possible, exception states shall be handled locally, in a manner that re-assimilates exception processing into the normal control flow. Unless required by program specifications, an Ada unit shall not output messages to the operator, abandon control to the operating system, or terminate execution.

5.11.4 Exception propagation. Exception states that cannot be handled locally shall be suitably recorded (e.g., flagged, counted, logged, filed) and propagated to the calling unit. The

Ada prolog shall identify and describe every Ada exception that can be propagated.

5.12 Generic units

5.12.1 Use of generic units. Provided that generic Ada code shall be subject to the same permissions and prohibitions as all other Ada code, the use of generic units shall be permitted:

a. To generalize package and subprogram units for use with different data types.

b. To pass subprograms or data types as parameters at invocation time.

c. To support software development with libraries of common objects, algorithms, and utilities.

d. When needed to comply with program specifications.

5.13 Implementation-dependent features

5.13.1 Use of dependent code. The use of machine-dependent or compiler-dependent Ada code shall be permitted:

a. To implement specified interfaces to devices, processes, objects, or code outside the Ada domain.

b. To meet performance or capacity constraints of the specified environment that cannot be met otherwise, such as speed or memory limits, transfer rates, and media capacities.

c. For systems programs that must operate near the hardware, firmware, operating system, or Ada environment level.

5.13.2 Format of dependent code. Machine-dependent or compiler-dependent Ada code shall:

a. Be isolated as an Ada package.

b. Be limited to the minimum code required.

c. Contain Ada comments identifying the requirements that justify its use.

d. Contain Ada comments documenting its effects.

COMMENTS/REVISIONS

Technical publications under the Information Resources Management (IRM) Standards and Guidelines Program (MCO 5271.1) are reviewed annually. Your comments and/or recommendations are strongly encouraged.

IRM Tech Pub Name: ______________________________

IRM-_____-___ (Number) Date of Tech Pub: ____________________

COMMENTS/RECOMMENDATIONS: ______________________________

Name/Rank: ______________________________ (optional)

Unit: ______________________________ (optional)

Mail To: Commandant of the Marine Corps (Code CCID-40)
Headquarters, United States Marine Corps
Washington, D.C. 20380-0001