



NATIONAL AERONAUTICS and SPACE ADMINISTRATION

Lyndon B. Johnson Space Center

Houston, Texas 77058

## SPACE SHUTTLE PROGRAMS

### ORBITER AVIONICS SOFTWARE PROGRAMMING <sup>STANDARDS</sup> DOCUMENT

Revision 4

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE).  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

PREPARED BY IBM UNDER  
NAS9-14444

Contract: NAS 9-14444

# SPACE SHUTTLE PROGRAMS

## ORBITER AVIONICS SOFTWARE PROGRAMMING STANDARDS DOCUMENT Revision 4

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED  
REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY




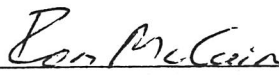
Federal Systems Division  
Houston, Texas


SPACE SHUTTLE  
ORBITER AVIONICS SOFTWARE  
NAS 9-14444

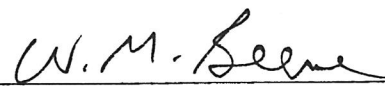
PROGRAMMING STANDARDS DOCUMENT  
SHUTTLE FLIGHT SOFTWARE  
REVISION 4

Technical Content Approval:

  
J. F. Clemons, Manager  
Avionics Software Development/  
Verification

  
R. C. McCain, Chairman  
Software Architecture Review Board

  
R. B. Ingenthron, Manager  
Systems Engineering

  
W. M. Beene, Project Manager  
Onboard Space Systems

IBM FEDERAL SYSTEMS DIVISION  
1322 Space Park Drive  
Houston, Texas 77058

82-SS-4556  
IRD No. 9a

TYPE II  
02/15/83

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

## PREFACE

This document constitutes a complete replacement for the "Orbiter Avionics Software Programming Standards Document (Revision 3)", published 2/2/82. All changes from that document are indicated with vertical bars in the right margin.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY



# TABLE OF CONTENTS

1.	INTRODUCTION
2.	NAMING CONVENTIONS
2.1	HAL/S IDENTIFIERS (M)
2.1.1	Label Names (M)
2.1.1.1	Code Block Labels (M)
2.1.1.1.1	PROGRAM, External Procedure and External FUNCTION Blocks (M)
2.1.1.1.2	COMPOOL Blocks (M)
2.1.1.1.3	Nested PROCEDURE, and FUNCTION Blocks (G)
2.1.1.2	Template Labels (M)
2.1.1.3	Executable Statement Labels (M)
2.1.2	Data Names (M)
2.1.2.1	Functional Data (M)
2.1.2.2	Local Non-Functional Data (G)
2.2	ASSEMBLER LANGUAGE IDENTIFIERS (M)
2.2.1	Label Names (M)
2.2.1.1	Code Block Labels (M)
2.2.1.1.1	CSECT and External Procedure Blocks (M)
2.2.1.1.2	Nested PROCEDURE and TASK Blocks (M)
2.2.1.2	Executable Statement Labels (G)
2.2.2	Data Names (M)
2.2.2.1	Functional Data (M)
2.2.2.2	Register Save Areas (M)
2.2.2.3	Local Non-Functional Data (G)
2.2.2.4	DSECTS (M)
2.2.2.4.1	DSECT Names (M)
2.2.2.4.2	Data Names In DSECTS (M)
2.2.2.5	HAL/S Data References (M)
2.2.2.5.1	Assembler COMPOOL Labels (M)
2.2.2.5.2	Assembler HAL/S Defined Data Entry Points (G)
2.2.2.5.3	Assembler I/O Buffer Entry Points (M)
2.2.3	Title Cards (G)
2.3	HAL/S INCLUDE SEGMENTS (M)
2.4	ASSEMBLER LANGUAGE COPY SEGMENTS (M)
2.5	DATA SET NAMES
2.6	DATA SET MEMBER NAMES (M)
2.6.1	HAL/S Generated Data Names (M)
2.6.2	Preprocessor and MACRO Generated Names (G)

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY MANNER, NOR PLACED IN ANY REPOSITORY

# TABLE OF CONTENTS (Cont'd)

3.	PROGRAMMING CONVENTIONS
3.1	TOP-DOWN PROGRAMMING (G)
3.2	CODE STRUCTURING (M)
3.2.1	Control Logic (M)
3.2.2	Code Block Size Limitation (G)
3.2.3	Flow Diagramming Standards for Crew Interface Grammar Statements (M)
3.3	SOFTWARE ANOMALIES (M)
3.3.1	Coding Restrictions (M)
3.3.1.1	Data Exchanges Between Processes
3.3.1.1.1	Data Exchanges Between Processes - Redundant Configurations (G)
3.3.1.1.2	Data Exchanges Between Processes - All Configurations (G)
3.3.1.2	Restrictions on EVENT Variables in Non R/T Statements (M)
3.3.1.3	FSW Process Priorities, Dates and Phase Off-Sets (M)
3.3.1.4	Redundant GPC Calculations (M)
3.3.1.5	Time Constraints on Software Sequences (M)
3.3.1.6	Restricted Use of HAL/S Real-Time Features
3.3.1.7	Characteristics of HAL/S SCHEDULE Statements (M)
3.3.1.8	Assembly Language Usage (M)
3.3.1.9	Constraints When Writing Disable Blocks/Exclusive Procedures (M/G)
3.3.1.9.1	Constraints When Writing Disable Blocks (M)
3.3.1.9.2	Constraints When Writing Exclusive Procedures (G)
3.3.1.10	Use of ON ERROR For I/O Operations (M)
3.3.1.11	Protected I/O Transactions (M)
3.3.1.12	Interprocess Variables Not Protected By Disable Blocks (M)
3.3.1.13	Processes Execution Over OPS Transitions (M)
3.3.1.14	Checksums (M)
3.3.1.15	RIGID COMPOOLS/STRUCTURES (M)
3.3.1.16	Local Data %COPY (M)
3.3.1.17	FSW Error Protection (M)
3.3.2	Coding Guidelines (G)
3.3.2.1	Removal of Unnecessary Diagnostic Messages (G)
3.3.2.2	Source Macro Definition Considerations (G)
3.3.2.3	FSW Data Referencing Considerations (G)
3.3.2.3.1	Optimization Considerations
3.3.2.3.2	Reliability Considerations
3.3.2.4	Restricted HAL Statements (G)
3.3.2.5	Data Initialization (G)
4.	DOCUMENTATION
4.1	SOURCE LISTINGS (M)
4.1.1	Prologue Comments (M)
4.1.2	Authorization ID Correlation Comments (M)
4.1.3	Statement Comments (G)
4.1.4	FSW Source Resequencing (M)
4.2	DESIGN SPECIFICATION FORMAT (M)
5.	REFERENCE MATERIAL

FOR RESEARCH USE ONLY

THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE).

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

## 1. INTRODUCTION

The Programming Standards Document For Shuttle Flight Software is provided to assist software development and verification personnel, at IBM FSD-Houston, in delivering a software package to NASA for the AP-101 computers onboard the Space Shuttle Orbiter Vehicle. This is accomplished by prescribing standards for the components of flight software development including naming conventions, program design, documentation, and program updating. References are made to documents that contain reference material for flight software personnel.

The flight software will largely be coded with the HAL/S language. Flight software development personnel coding in any other language will still be expected to follow the standards of this manual as closely as possible. These standards and additional data and baseline information provided in Software Awareness Memos (SAMs) are either Mandatory (M) or Guidelines (G). Each document or paragraph will be so designated and each element will be assumed to be the designated level unless explicitly stated.

Exceptions to mandatory standards/SAMs must be approved by the Software Architecture Review Board (SARB) and will be documented as part of SAM 2. Exceptions to guidelines may be approved by the design/code inspection team only if a superior alternative is demonstrated. Guideline exceptions will be documented as part of the retained design/code review package.

Changes to this document, or to SAM documentation, may be requested by submitting a FAIR (see Attachment A) to a SARB member. Appropriate updates will be made after review and disposition by the SARB.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

## 2. NAMING CONVENTIONS

The naming conventions for identifiers, include segments, data sets, and data set members, are documented in this section. These standards are designed to enable generation of unique identifiers and data set names for each area with little interfacing. Standards relating to HAL/S and Assembler language are discussed in separate subparagraphs. Mandatory (M) or Guidelines (G) is indicated on each paragraph.

### 2.1 HAL/S IDENTIFIERS (M)

The naming conventions for the HAL/S identifiers are specified in the following paragraphs for either label or data names. In generating label and data names the following rules must be adhered to:

- o The total number of characters in a single name must not exceed 32. The total number of characters in a fully qualified name may not exceed 50. It is recommended however, that all symbol names be defined concisely for readability and to maximize processing and storage efficiency.
- o The first character must be alphabetic and any character after the first may be alphabetic or numeric.
- o Any character except the first or the last may be a 'break character' ( ) (except explicit formats stated in subparagraphs).
- o Control Segment Grammar Keywords cannot be used as identifiers (See Table 2-1).
- o HAL/S keywords and built-in function names cannot be used as identifiers (See HAL/S-FC User's Guide).

#### 2.1.1 Label Names (M)

This section specifies the naming conventions established for labels on HAL/S code blocks, templates, and executable statements. The formats for assigning names to these labels are discussed in detail in the following paragraphs.

Variations of the conventions are presented in Section 2.2 for the AP101 Assembly Language code and data blocks.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY

#### 2.1.1.1 Code Block Labels (M)

Three formats were established for assigning names to labels on code blocks. These formats are defined for the different types of code blocks which can be generated by HAL/S. The first format defined is for labels on PROGRAM, external FUNCTION and external PROCEDURE code blocks. The second format is for labels on COMPOOL code blocks. The third format is for labels on nested PROCEDURE, FUNCTION, UPDATE and TASK code blocks.

Downlist units should start with a 'DCD' prefix and display units names must start with a 'C' and the fourth character must not be an underscore. This standard is defined to accommodate automatic determination of display/downlist units by the IPV Analysis PGM.

Macro-defined and preprocessor generated labels may include additional prefixes (paragraph 2.6.2.4). The linkage editor requires that the first six characters (not counting break characters) be a unique combination for all unit of compilation labels relative to each code block type. This constraint is met since uniqueness is maintained for the first three characters, according to the definitions presented in the following sub-paragraphs.

##### 2.1.1.1.1 PROGRAM, External PROCEDURE and External FUNCTION Blocks (M)

The format for PROGRAM, external PROCEDURE and external FUNCTION Block label is:

ABB\_C...C

Where:

- A - Flight Software Subsystem ID (Functional Area)
  - = G (Guidance, Navigation, & Control (GN&C))
  - = A (System Control (SC))
  - = D (User Interface (UI))
  - = P (Payload (PL))
  - = S (Systems Management (SM))
  - = V (Vehicle Checkout (VCO))
  - = R (Remote Manipulator System)
- BB - Unique (relative to the subsystem) alphanumeric ID assigned by the designated programmer(s) for each software subsystem
- \_ = Break character

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

C...C - Alphanumeric ID descriptive of the purpose of the code block. This ID must be coordinated with the designated programmer(s) prior to assignment (maximum of 28 characters).

#### 2.1.1.1.2 COMPOOL Blocks (M)

The format for COMPOOL Block labels is:

BAD\_C...C

- Where:
- B - C for Flight Software Subsystems other than FCOS
  - A - Flight Software Subsystem ID described in paragraph 2.1.1.1.1. In addition, 'Z' may be used for COMPOOLS that are associated with multiple flight software subsystems.
  - D - Alphanumeric ID assigned by the designated programmer(s) for each software subsystem.
  - Break character
  - C...C - Alphanumeric ID assigned by the designated programmer(s) (maximum of 28 characters).

#### 2.1.1.1.3 Nested PROCEDURE and FUNCTION Blocks (G)

The format for nested PROCEDURE, FUNCTION, and TASK Block labels is:

ABB\_C...C

- Where:
- A - Flight Software Subsystem ID, assigned in paragraph 2.1.1.1.1.
  - BB - A unique or same "BB" as defined in paragraph 2.1.1.1.1.
  - Break character
  - C...C - Alphanumeric ID descriptive of the purpose of the code block. This ID must be unique within a program, external function or external procedure code block (maximum of 28 characters).

FOR RESEARCH USE ONLY

THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES

SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY MANNER, NOR PLACED IN ANY REPOSITORY

#### 2.1.1.2 Template Labels (M)

There are four kinds of code templates: PROGRAM, PROCEDURE, FUNCTION and COMPOOL. The label of each template is the same as the label of the corresponding code block.

#### 2.1.1.3 Executable Statement Labels (M)

No specific naming conventions are specified for Executable Statement Labels. However, for ease of testing, the CLOSE statement for code blocks must be labeled with the same three-character ID described in section 2.1.1.1.1 followed by an underscore and the word CLOSE as shown below:

ABB\_CLOSE

#### 2.1.2 Data Names (M)

There are two types of data which will be used in the Flight Software programs:

- o Functional Data

The availability of the HALSTAT listing from the HAL/S compiler makes it unnecessary to define a separate convention of COMPOOL data as in the past. Therefore, to aid in design flexibility (i.e., the flexibility to move data from local store to COMPOOL and vice versa), a convention for Functional Data is established.

- o Functional Data is data which is controlled by a functional area (reference 2.1.1.1.1) and can be DECLARED locally or as COMPOOL data. Functional Data must be in a COMPOOL if it is shared by two or more compilation units, otherwise it can be DECLARED locally. Items passed as CALL arguments need not be in a COMPOOL. Replace names, Structure templates, and all levels of qualified names need not have a prefix (reference 2.1.2.1). Only one qualification level must have a prefix. Filler data and data defined in a display or AMT COMPOOL need not adhere to functional data naming standards.

- o Local Data

Local Data is data which is truly local and will never be shared by two or more compilation units (e.g., loop counters, indices, subscripts, intermediate storage, etc.). No conventions are established for this data.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY

BOOK: Programming Standards

2.1.2.1 Functional Data (M)

The format for Functional Data is:

BADN\_Y...Y

Where:

- B - C for Flight Software Subsystems other than FCOS
- A - Flight Software Subsystem id, described in paragraph 2.1.1.1.1. Same as corresponding ID for the defining compilation unit.
- D - Alphanumeric character assigned by the designated programmer(s) for each software subsystem describing the function responsible for the parameter. As a guideline, same as corresponding alphanumeric character for defining COMPOOL if data item within COMPOOL.
- N - The standard for this character is a guideline. One of the following characters should be used as the fourth character. Only an applicable identifier should be selected and they should be selected in the priority indicated if multiple options apply:
  - K - declared CONSTANT
  - E - declared EVENT variable
  - B - declared BOOLEAN, BIT variable
  - V - declared VARIABLE but not BIT, BOOLEAN, EVENT
- Break character
- Y...Y - Alphanumeric ID which must be unique within the Functional area. This field should be descriptive and, as a goal, should be limited to approximately ten characters.

2.1.2.2 Local Non-Functional Data (G)

No conventions are established for naming of local data; however, these labels should not be a form that would be confused with the functional data.

FOR RESEARCH USE ONLY  
 THIS MATERIAL MAY BE  
 PROTECTED BY COPYRIGHT LAW  
 (TITLE 17 U.S. CODE)  
 COPY PROVIDED BY  
 WICHITA STATE UNIVERSITY LIBRARIES  
 SPECIAL COLLECTIONS  
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
 REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY



## 2.2 ASSEMBLER LANGUAGE IDENTIFIERS (M)

The naming conventions for the AP101 Assembler identifiers are specified in the following paragraphs for either label or data names. In generating label and data names the following rules must be adhered to:

- o The total number of characters must not exceed 8.
- o The first character must be alphabetic except for the special characters required to match the names of compilation units of HAL/S (Section 2.6.1 and 2.6.2), and those generated by the structured programming macro statements. Any character after the first may be alphabetic or numeric.

### 2.2.1 Label Names (M)

This section specifies the naming conventions established for labels on Assembler code blocks and executable statements. The formats for assigning names to these labels are discussed in detail in the following paragraphs.

All labels of application (i.e., non-FCOS) control sections (CSECTs) must have two special characters preceding the unique prefix discussed under HAL/S. These characters must correspond to those generated by HAL/S (Section 2.6.1 and 2.6.2).

In addition, the building of similar language library routines require the following prefix:

Library	-	AB; where A = (A-Z), B = (A-Z)
Library ZCONS	-	#Q
Sector Zero	-	#O
Library Data	-	#L

#### 2.2.1.1 Code Block Labels (M)

Two formats were established for assigning names to labels on code blocks. These formats are defined for the different types of code blocks which can be generated. The first format defined is for labels on CSECT and external PROCEDURE code blocks. The second format is for labels on nested PROCEDURE and task code blocks.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY

#### 2.2.1.1.1 CSECT and External PROCEDURE Blocks (M)

The format for CSECT and external PROCEDURE block labels is:

FAABBBBB

Where: F = Flight Computer Operating System (FCOS)

AA = CM (Configuration Management)  
= IO (I/O Management)  
= PM (Process Management)

B...B = Alphanumeric ID descriptive of the purpose of the code block. This ID must be coordinated with the designated programmer(s) prior to assignment (maximum of 5 characters)

#### 2.2.1.1.2 Nested PROCEDURE and Task Blocks (M)

The format for nested PROCEDURE and task block label is:

FAABBBBB

Where: F = Flight Computer Operating System (FCOS)

AA = Same AA as defined in paragraph 2.2.1.1.1

B...B = Alphanumeric ID descriptive of the purpose of the code block. This ID must be unique within a program or external procedure code block (maximum of 5 characters).

#### 2.2.1.2 Executable Statement Labels (G)

No specific naming conventions are specified for Executable Statement Labels.

#### 2.2.2 Data Names (M)

There are two types of data which will be used in the Flight Software programs:

##### o Functional Data

Functional data is data which is controlled by a functional area and is declared in a general data CSECT designed to gather this data in one place. Normally the data field is required for use by more than one executable CSECT and is referenced by name rather than passed as a parameter. This data also shares the need to be defined in a data sector.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY

o Local Data

Local Data is data which is truly local and will never be shared by two or more compilation units (e.g., loop counters, indices, subscripts, intermediate storage, etc.). No conventions are established for this data.

2.2.2.1 Functional Data (M)

The format for Functional Data is:

FAAB...B

Where: F = FCOS

AA = CM (Configuration Management)  
= IO (I/O Management)  
= PM (Process Management)

B...B - Alphanumeric ID which must be unique within the Functional area. This field should be descriptive and must be limited to five characters.

2.2.2.2 Register Save Areas (M)

The format for areas reserved to save a set of registers upon entry to an FCOS CSECT is:

FAAB...B

Where: F = FCOS

AA = C\$ (Configuration Management)  
= I\$ (I/O Management)  
= P\$ (Process Management)

B...B = Alphanumeric ID which is normally the same as the CSECT the save area is reserved for (limited to five characters).

2.2.2.3 Local Non-Functional Data (G)

No conventions are established for naming of local data.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

Date: 02/02/82

Rev: 3

Page 2-10

BOOK: Programming Standards

2.2.2.5 HAL/S Data References (M)2.2.2.5.1 Assembler Compool Labels (M)

The format of Compool block labels in compools generated for assembler language program to HAL/S program interface (e.g., FCMCOM data) is:

TFAAB...B

Where: TF = 'TF'

AA = CM (Configuration Management)  
 = IO (I/O Management)  
 = PM (Process Management)

B...B = Alphanumeric ID descriptive of the purpose of the data value. This ID must be coordinated with the designated programmer(s) prior to assignment (maximum of 4 characters).

2.2.2.5.2 Assembler HAL/S Defined Data Entry Points (G)

The format for assembler required HAL/S data entry points other than I/O buffers is:

AAAAB...B

Where: AAAA = The first four characters of the HAL/S data name

B...B = Alphanumeric ID assigned by the designated programmer(s) (maximum of 4 characters).

2.2.2.5.3 Assembler I/O Buffer Entry Points (M)

The format for assembler entry points for I/O buffers is:

TFAAB...B

Where: TF = Fixed characters indicating an FCOS referenced label

AA = IV (Input I/O buffer)  
 = OV (Output I/O buffer)

B...B = Alphanumeric ID assigned by the designated programmer(s) (maximum of 4 characters)

FOR RESEARCH USE ONLY  
 THIS MATERIAL MAY BE  
 PROTECTED BY COPYRIGHT LAW  
 (TITLE 17 U.S. CODE)  
 COPY PROVIDED BY  
 WICHITA STATE UNIVERSITY LIBRARIES  
 SPECIAL COLLECTIONS  
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
 REPRODUCED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPYING, RECORDING, OR BY ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM.

Date: 02/02/82

Rev: 3

Page 2-9

BOOK: Programming Standards

#### 2.2.2.4 DSECTS (M)

##### 2.2.2.4.1 DSECT Names (M)

DSECT names are in the following format:

TAX...X

Where: T = 'T'

A = An optional field which if present is an F  
for FCOS

X...X = one to four characters that uniquely  
identify the table

##### 2.2.2.4.2 Data Names in DSECTS (M)

The names of data fields in DSECTS are in the following format:

TX...XY...Y

Where: T = 'T'

X...X = The first three characters of the  
X...X field in the DSECT name

Y...Y = one to four characters which uniquely  
identify the data field

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY MANNER NOR PLACED IN ANY REPOSITORY

### 2.2.3 Title Cards (G)

Each block of code will include a TITLE card which provides the name (eight character limit) and a longer descriptive name for the block. The form is:

name TITLE 'complete name or title of block'

### 2.3 HAL/S INCLUDE SEGMENTS (M)

Include segments are blocks of application source code stored as PDS (partitioned data set) members. These segments can be included at compile time as part of a program, external procedure, or external function by using the INCLUDE compiler directive. The include segment will be compiled as if the segment had been coded in line. The following format will be used for naming all include segments, except system level macro sequences:

ABBC...C

Where:

- A - Flight Software Subsystem ID described in paragraph 2.1.1.1.
- BB - A same or unique 'BB' as defined in paragraph 2.1.1.1.1.
- C...C - Alphanumeric ID descriptive of the purpose of the code block. This ID must be coordinated with the designated programmer(s) prior to assignment (maximum of five characters).

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE).  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY.

## 2.4 ASSEMBLER LANGUAGE COPY SEGMENTS (M)

COPY segments are blocks of source code stored as partitioned data set (PDS) members. These segments can be copied by using the COPY assembler directive at assembly time. The copied segments will be assembled as if the segments had been coded in line.

The following is the format for naming all copy segments:

FAAB...B

Where: F = Flight Computer Operating System (FCOS)

A = CM (Configuration Management)

= IO (I/O Management)

= PM (Process Management)

B...B = Alphanumeric ID descriptive of the purpose of the code block. This ID must be coordinated with the designated programmer(s) prior to assignment (maximum of five characters).

## 2.5 DATA SET NAMES

FSW data set naming conventions are a joint responsibility of the Flight Software Integration Team (FIT) and build group.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

## BOOK: Programming Standards

## 2.6 DATA SET MEMBER NAMES (M)

The names for data set members will be generated for flight software by either the HAL/S Compiler or the programmer creating the member. The HAL/S Compiler will generate member names of object code, templates and simulation data files for the compilation units. The programmer will assign member names for the source data sets. Names must be registered with the build coordinator, and certain information supplied before the member can become part of the system.

FSW source member names will be precisely the same as the corresponding non-underscore characters of the code block name. As a guideline, non-FCOS member names should be constrained to the first 6 non-underscore characters of the code block name.

2.6.1 HAL/S Generated Names (M)

The HAL/S Compiler will generate the member name for the object code, template and simulation data file for a compilation unit in the following manner:

HAL/S compilation unit names are transferred to the emitted object code, by using only the first six characters of the HAL/S name. Any occurrence of the underscore character ( ) in the first six characters of a TASK, PROGRAM, PROCEDURE, FUNCTION, or COMPOOL name is eliminated. The resulting characters are joined together to produce the name of the compilation unit (e.g., A\_B\_C becomes ABC). An additional two characters are placed on the front of the resultant name to form a unique control section name for each of the individual situations in which the name is used. CSECT naming conventions are shown in Figure 2.6.1-1.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY



Date: 11/15/82

Rev: 4

Page 2-13a

BOOK: Programming Standards

<u>CSECT Type</u>	<u>Primary Name</u>	<u>Overflow Name</u>
CODE CSECTS:		
Program	\$ONNNNNN	\$VNNNNNN
Tasks	\$cNNNNNN c=(1-F) for a limit of 15 tasks	\$cNNNNNN c=(G-U)
COMSUBs	#CNNNNNN	\$WNNNNNN
Internal Procedures	anNNNNNN a=(A-M) n=(0-9) for a limit of 130 procedures	bnNNNNNN b=(N-Z)
Libraries	aaNNNNNN a=(A-Z)	
Patch-User defined	\$Yaab000 aa=phase, b=sector \$Yaab001	
Patch-from MM Build	\$Yaabnnn aa=phase, b=sector, nnn > 1.	
DATA CSECTS:		
Stack	@cNNNNNN c=(0-9, A-F)	
DECLARE Data	#DNNNNNN	#SNNNNNN
REMOTE Data	#RNNNNNN	#UNNNNNN
COMPOOL Data	#PNNNNNN	#VNNNNNN
Patch-User defined	#Yaab000 aa=phase, b=sector #Yaab001	
Patch-From MM build	#YaaBnnn aa=phase, b=sector, nnn > 1.	
OTHER CSECTS:		
ZCON to COMSUB	#ZNNNNNN	
ZCON for library routine	#ONNNNNN	
Bank Zero	#ONNNNNN	
Process Directory Entry	#ENNNNNN	
Data for Library Routine	#LNNNNNN	
EXCLUSIVE Data	#XNNNNNN	#WNNNNNN
Pad Space for ZCONs/QCONs	\$Xaannnn aa=Phase nnnn=Pad no.	
OTHER NAMES:		
Support Data Files	##NNNNNN	
Templates	@NNNNNN	

Figure 2.6.1-1 CSECT Naming Conventions

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

In addition to control section names, the name of a compilation unit may appear in certain external contexts preceded by the following characters:

- @@ - the member name of the template created for a compilation unit.
- ## - the member name of the simulation data file created for a compilation unit.

#### 2.6.2 Preprocessor and MACRO Generated Names (G)

Symbols may be generated that follow the conventions of the functional area involved (i.e., subsystem ID defined in section 2.1) or have one of the following prefix characters:

- M - DFG Preprocessor (Displays)
- Z - System Level Source Macro (e.g., Disable/Enable)
- XX... - Any prefix that corresponds to prefix of source member (e.g., XX...MACS)

Additional symbols may be reserved if they increase readability (see Table 2-1 as an example). User's Guides will define additional sets of symbols, unique to a particular Macro Source Member and function.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY MANNER WITHOUT THE WRITTEN PERMISSION OF THE  
LIBRARY

TABLE 2-1

CREW INTERFACE GRAMMAR KEYWORDS

ADVANCE_ON	ITEM_EXEC
BLOCK	ITEM_I
BLOCK_CLEAN_UP	ITEM_NO
BLOCK_END	ITEM_O
CHANGE	ITEM_S
CLEAN_UP	KEY
CLEAN_UP_MODE	MODE
CLEAN_UP_OPS	MODE_CLEAN_UP
CLEAN_UP_SPEC	MODE_END
D_BLOCK_NUMBER	NO_AUTO_ADVANCE
D_DEU_NUMBER	NO_CLEAN_UP
D_IND	NO_CLEAN_UP_MODE
DISPLAY	NO_CLEAN_UP_OPS
D_MODE_NUMBER	NO_CLEAN_UP_SPEC
D_NEW_MODE_NUMBER	OPS
N_NEW_OPS_NUMBER	OPS_CLEAN_UP
D_OPS_NUMBER	OPS_END
DS	PRO
D_SPEC_NUMBER	RESUME
EXEC	SPEC
INIT_BLOCK	SPEC_CLEAN_UP
ITEM_ENTER	SPEC_END

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY

### 3. PROGRAMMING CONVENTIONS

Program implementation will follow the rules of top-down, structured programming (see Reference Material) in order to optimize the ability of programmers to develop and modify flight software. It is assumed that the flight software will be coded in the HAL/S language which is designed to implement the features of top-down, structured programming. Mandatory (M) or Guideline (G) is indicated on each paragraph.

#### 3.1 TOP-DOWN PROGRAMMING (G)

The top-down programming technique is applicable to software development at the system level and at the module level.

At the system level, the basic control modules are coded first, then the first level of application modules is coded while the basic level is checked out, invoking dummy first level application modules. Each level is developed in this manner, according to successive levels of functional detail, down to the most detailed modules.

Within each module, the top-down approach is used by coding the nucleus, or top level, of control code first and adding sections of functional code in the order of their level of detail. If possible, the nucleus should be simply a string of references to external (FUNCTIONs, PROCEDURES, PROGRAMs, and include-segments) and internal (TASKs, FUNCTIONs and PROCEDURES) code blocks which at first need only to return immediately to the module's nucleus. While the nucleus is being verified, the first level of referenced code blocks should then be coded even though it may also reference dummy code blocks at first. Thus each successively more detailed level of referenced code blocks is developed until the entire module satisfies all of its detailed specifications.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY MANNER IN ANY REPOSITORY

## 3.2 CODE STRUCTURING (M)

Code structuring standards in a structured programming environment include provisions for:

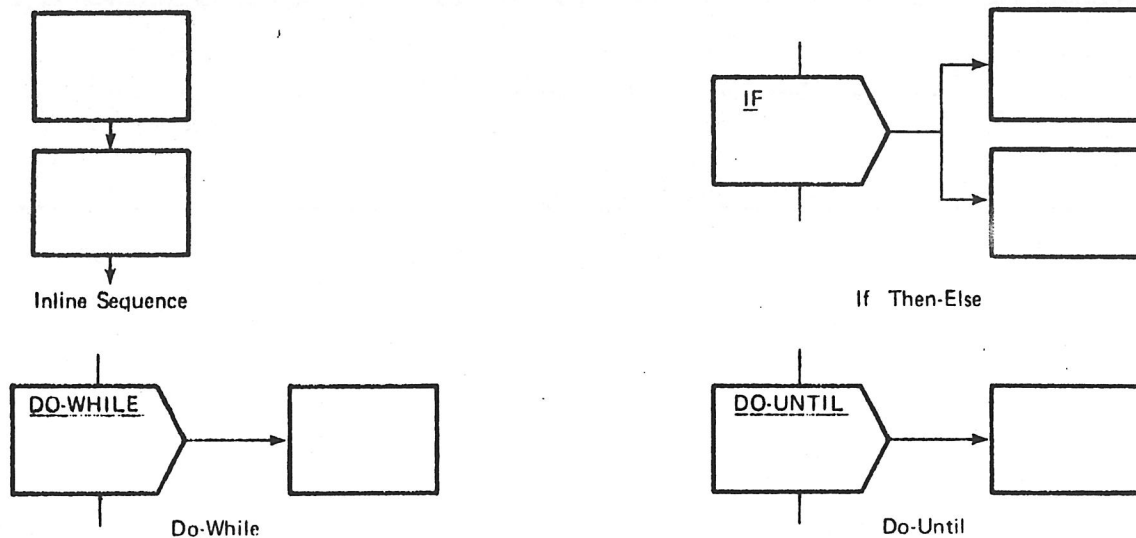
1. Single entry and single exit point for each code block (not applicable to FCOS).
2. Return to the next executable statement following the calling statement when a procedure or function is invoked (not applicable to FCOS).
3. Simplified control logic.

The standards for items 1 and 2 are controlled by the HAL/S restrictions for code blocks. Each code block may have only one header statement and one CLOSE statement. The CLOSE statement terminates the code block and returns execution to the first executable statement after the calling statement. Exceptions can be made to items 1 and 2, to provide efficient handling of abnormal code block termination situations. These exceptions must be approved by the SARB.

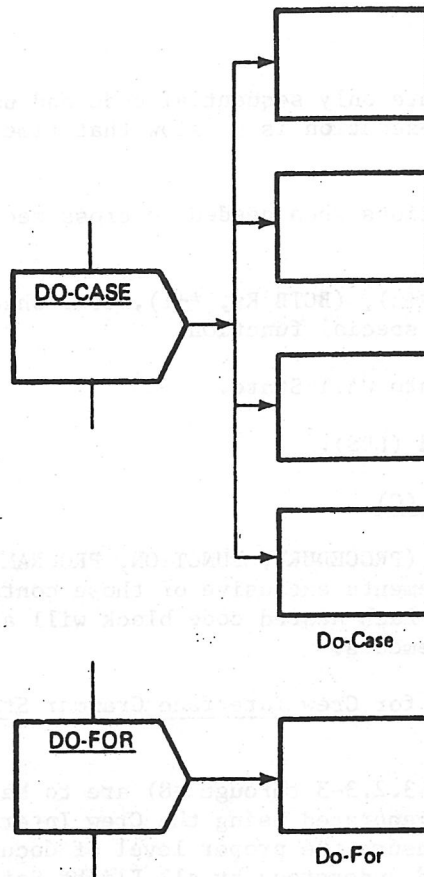
The standards for item 3 are programmer controlled and will be discussed in the following paragraphs.

3.2.1 Control Logic (M)

Structured programming enhances readability and maintainability of code, and the control logic should reflect these attributes. Thus, the flight software will use only the following structured constructs for code sequencing:



FOR RESEARCH USE ONLY  
 THIS MATERIAL MAY BE  
 PROTECTED BY COPYRIGHT LAW  
 (TITLE 17 U.S. CODE)  
 COPY PROVIDED BY  
 WICHITA STATE UNIVERSITY LIBRARIES  
 SPECIAL COLLECTIONS  
 WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
 REPRODUCED IN ANY MANNER, NOR PLACED IN ANY REPOSITORY



FOR RESEARCH USE ONLY  
 THIS MATERIAL MAY BE  
 PROTECTED BY COPYRIGHT LAW  
 (TITLE 17 U.S. CODE)  
 COPY PROVIDED BY  
 WICHITA STATE UNIVERSITY LIBRARIES  
 SPECIAL COLLECTIONS  
 WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
 REPRODUCED IN ANY FORM, NOR PLACED IN ANY REPOSITORY

The Inline Sequence represents contiguous statements being executed sequentially. The If-Then-Else logic chooses one of two paths based on the result of a binary comparison. The Do-While logic repeats execution of a statement or group of statements as long as the test condition is true (i.e., the test precedes each cycle of execution). The Do-Until logic repeats execution of a statement or group of statements until the test condition becomes true (i.e., the test follows each cycle of execution). The Do-Case logic chooses one of at least three possible paths based on the result of a sequential search through an index list. The Do-For repeats the process block according to the user specification. Any code that does not adhere to the above structured constructs, including usage of GO TO or assembler language explicit branches, must be approved by the SARB. An approved exception is for the HAL/S EXIT statement.

The following NONHAL exceptions are approved:

- o Macro library (MLIB80) code segments used to implement the structured macros.
- o Sync routines (for efficiency in main loop timing).

## BOOK: Programming Standards

- o IOP code (BCEs can execute only sequential code and unconditional branches; MSC execution is so slow that time is critical).
- o Explicit branch instructions when needed to cross sector boundaries.
- o (BALR Rx, 0), (BAL Rx, \*+2), (BCTB Rx, \*+1), etc. which do not branch but perform some special function.
- o Set System Mask (SSM) into WAIT State.
- o Load Program Status Word (LPS).

### 3.2.2 Code Block Size Limitation (G)

Each HAL/S source code block (PROCEDURE, FUNCTION, PROGRAM) will be limited to approximately 100 statements exclusive of those contained in nested code blocks and comments. Each nested code block will also be limited to approximately 100 statements.

### 3.2.3 Flow Diagramming Standards for Crew Interface Grammar Statements (M)

Standard flow forms (Figures 3.2.3-3 through -8) are to be used in documenting all control segments generated using the Crew Interface Grammar Statements. This is to insure the proper level of documentation occurs and is universally used and understood by all Flight Software programmers.

Figures 3.2.3-1 and 3.2.3-2 denote the generic forms for the UI grammar statements for operational sequences and specialist functions respectively. The circled references are for presentation purposes only. These references correlate the generic form of the grammar statement and the figure and block where the standard form is denoted.

Figures 3.2.3-3, 5 and 7 are overviews of OPS, MODE and BLOCK selection logic. The decisions and sequencing portrayed in these figures are largely controlled by the System Services transition matrices as well as crew selection through the MCDS, sequencing events and CHANGE grammar statements. All blocks not explicitly representative of applications code are to be indicated by an asterisk.

Figure 3.2.3-3 is to identify all SPEC's valid in the major function (via the standardized table form) as well as all OPS and associated OPS sequencing. All blocks in this overview will be flagged as representative by an asterisk.

Figure 3.2.3-5 identifies mode within an OPS and associated mode sequencing. Again an asterisk will appear in each block.

Figure 3.2.3-7 similarly depicts the block sequencing within a mode. Each block would contain an asterisk.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

The following are to be followed:

1. Flow standards are denoted via capital letters, unique text is denoted via script.

Example: IF condition\_k

2. Optional code inclusions are bracketed.

Example: IF NEW MODE OR event\_1

3. Lower level flows may be included within a higher level so long as the upper level doesn't get too cluttered.
4. The DO UNTIL form within a BLOCK may be eliminated providing:
  - o The BLOCK MODE nor OPS statement had automatic advancement event specified.
  - o There is always a CHANGE statement executed following the processing of any MCDS input passed to the control segment.
5. The DO UNTIL form within a MODE may be eliminated providing:
  - o There is only one MODE within the OPS.
  - o No automatic advancement on events was specified for the MODE.
6. Transitions from/to Systems Services OPS-0 should not be shown in application flows. This will be shown in a level above the major function overview.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY



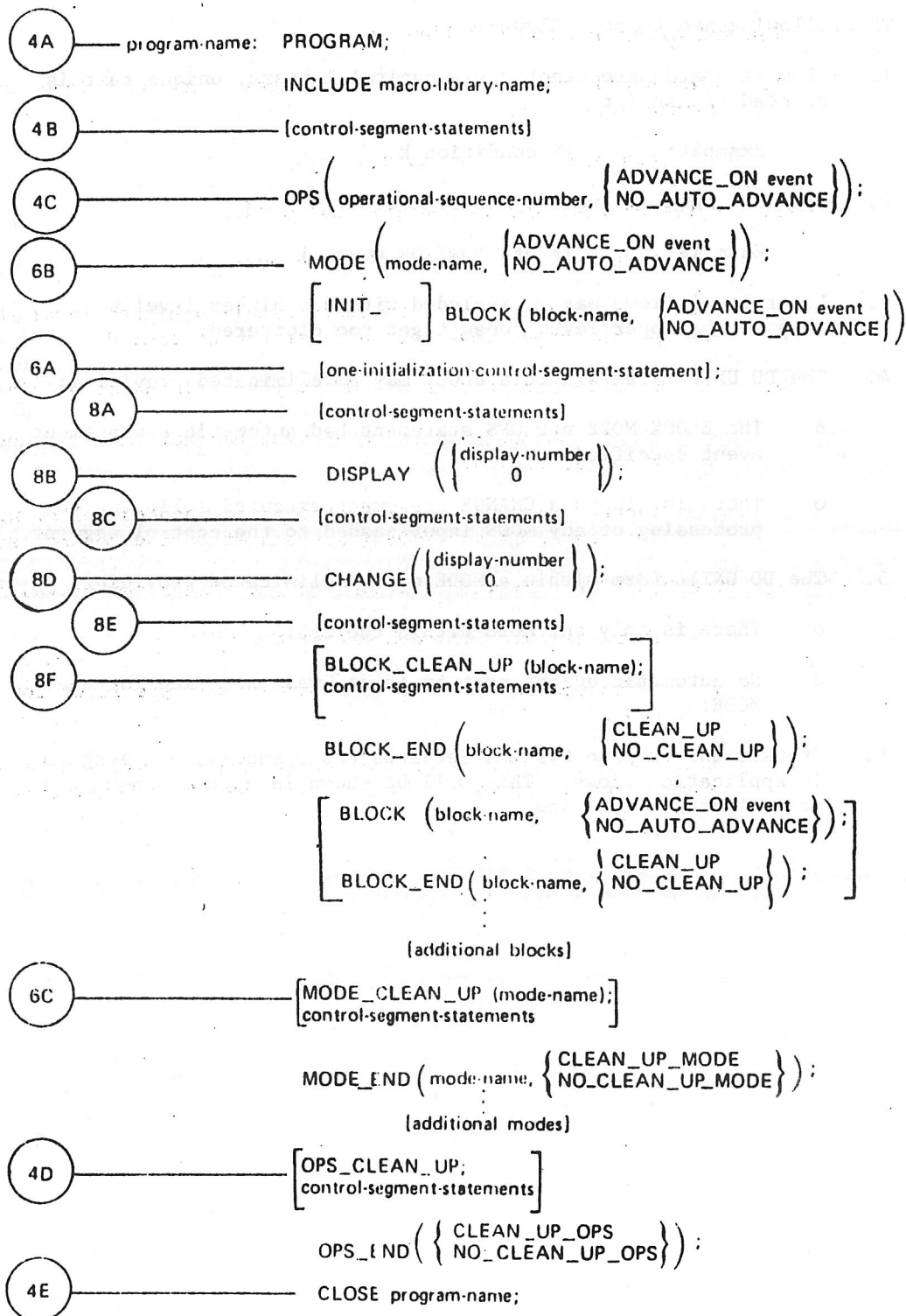
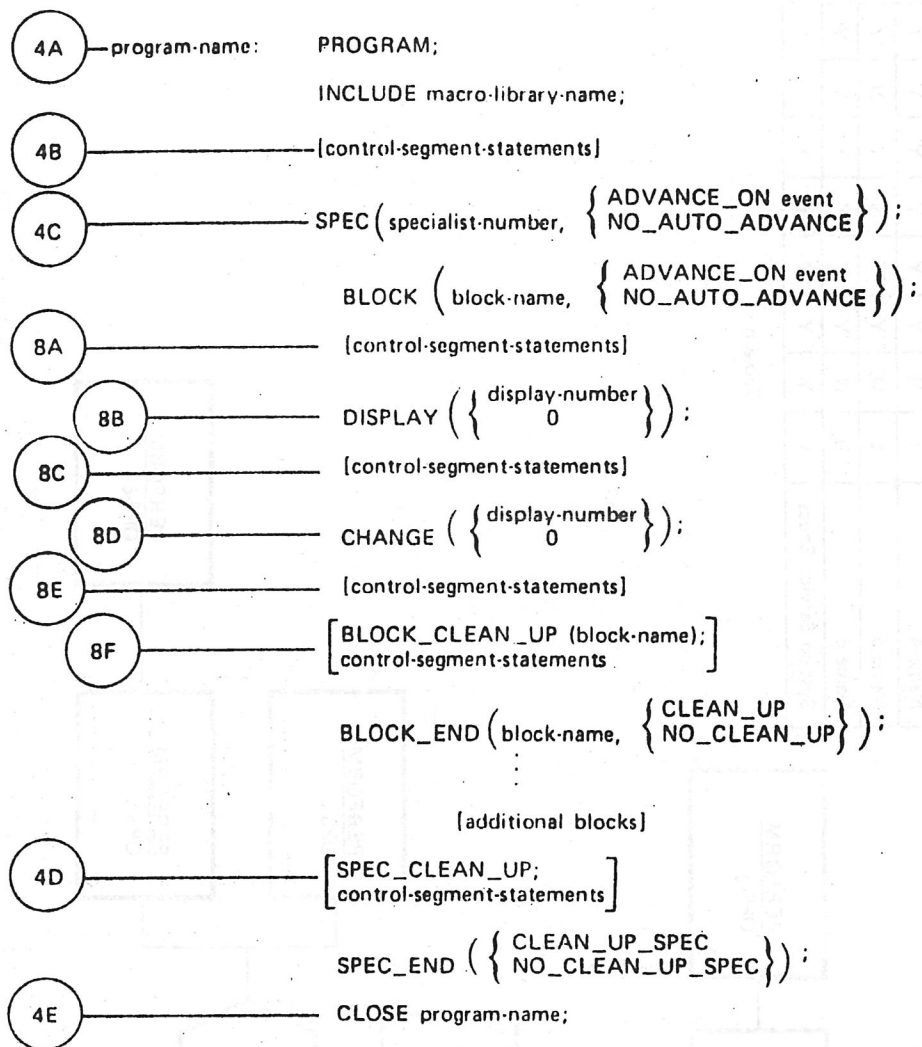


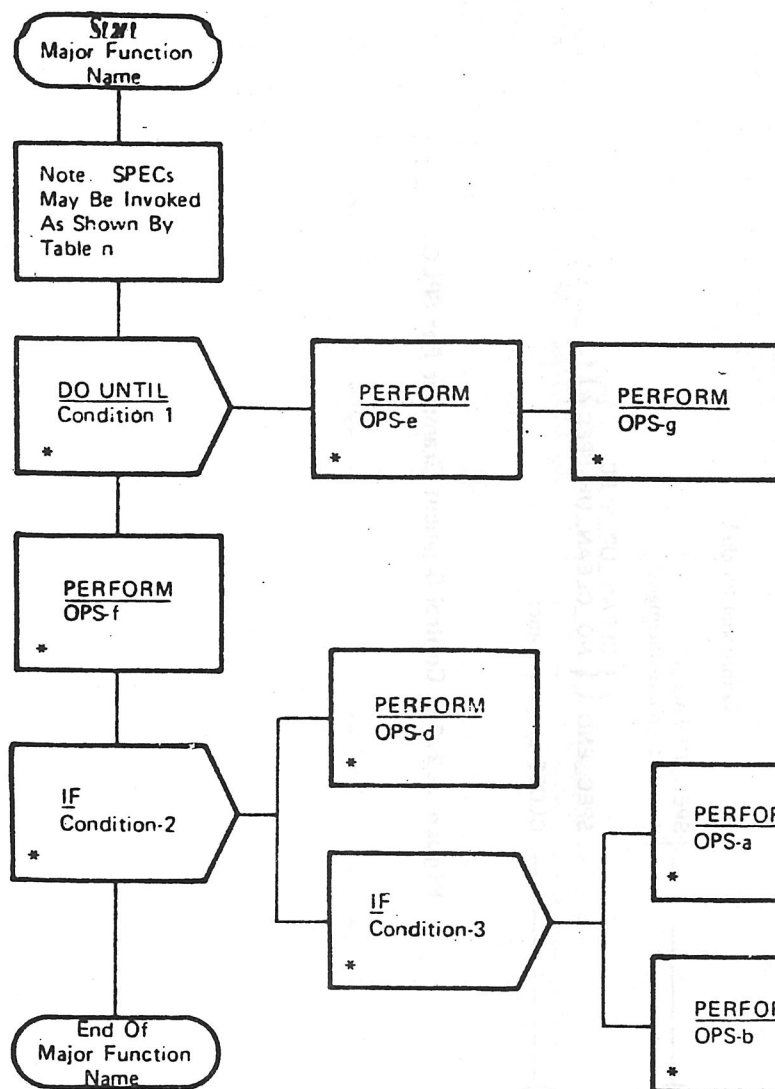
Figure 3.2.3-1 Control Segment Grammar For OPS

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY



FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION. THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

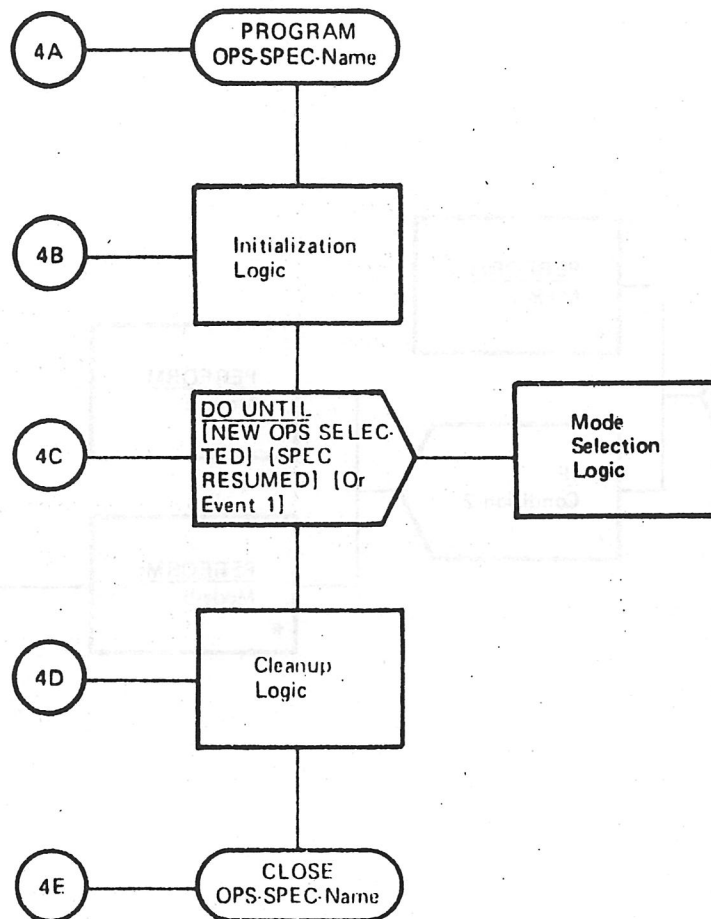
Figure 3.2.3-2 Control Segment Grammar For SPEC



SPEC Function	Flow Page	Valid In OPS						
		a	b	c	d	e	f	g
Name-1	p	Y	N	N	N	Y	N	Y
Name-2	q	N	Y		N	N	Y	Y
Name-3	r	N	N	Y	N	Y	Y	N
Name-4	s	N	Y	Y	Y	Y	Y	Y
Name-5	t	N	Y	Y	N	Y	N	Y
Name-6	u	N	Y	Y	Y	Y	Y	Y
System Service Specs	v	Y	Y	Y	Y	Y	Y	Y

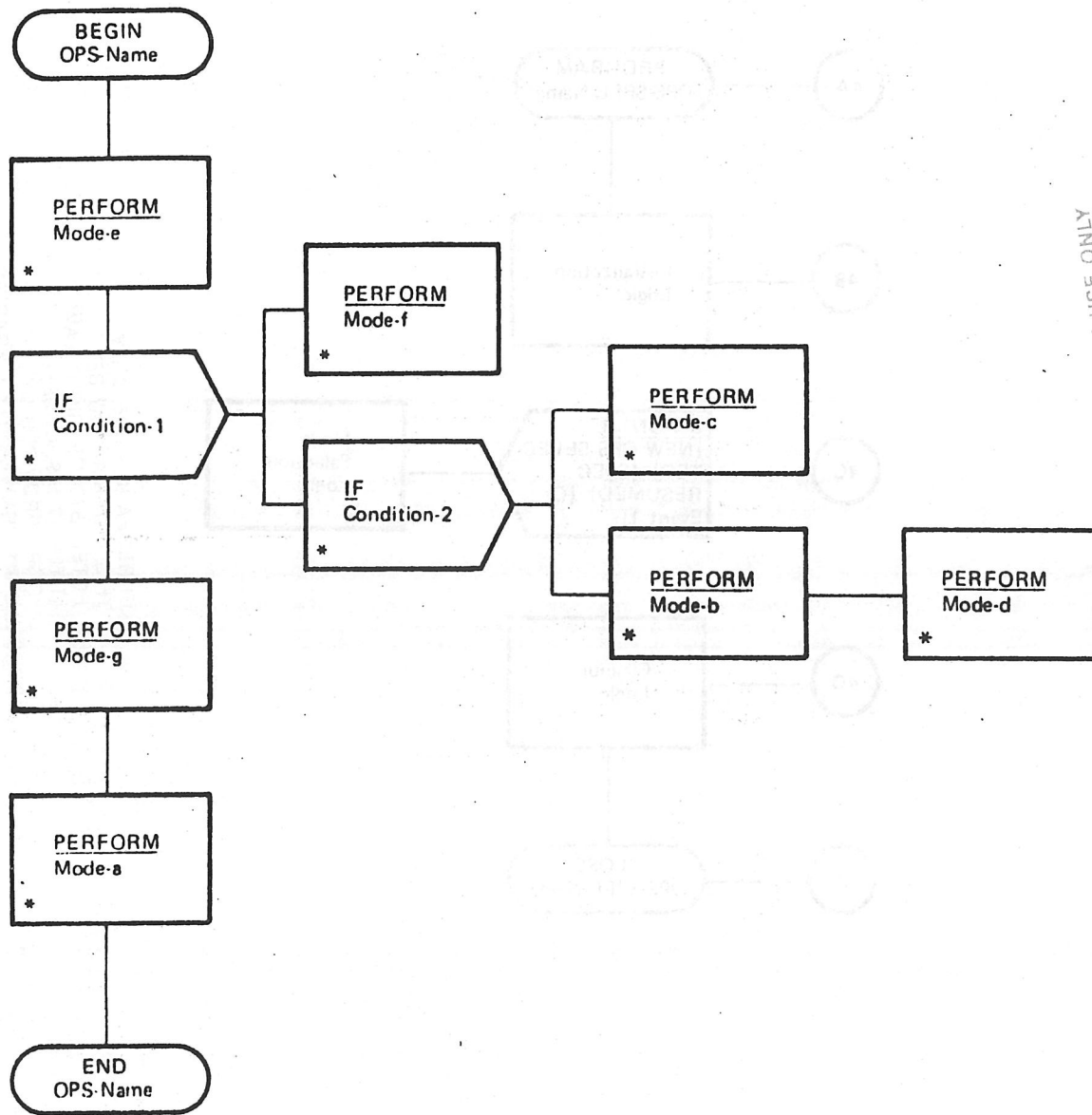
Table n

Figure 3.2.3-3 Major Function Name Overview



FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

Figure 3.2.3-4 OPS/SPEC Name Overview



FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

Figure 3.2.3-5 OPS Name SEQUENCING

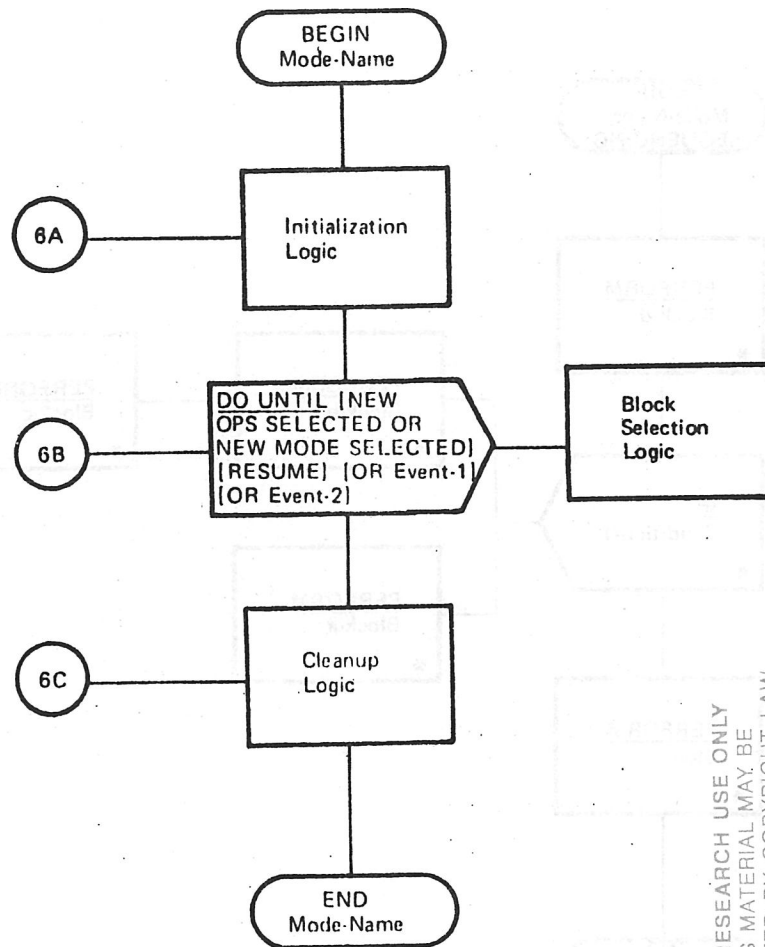


Figure 3.2.3-6 MODE Name OVERVIEW

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE).

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY MANNER, NOR PLACED IN ANY REPOSITORY.

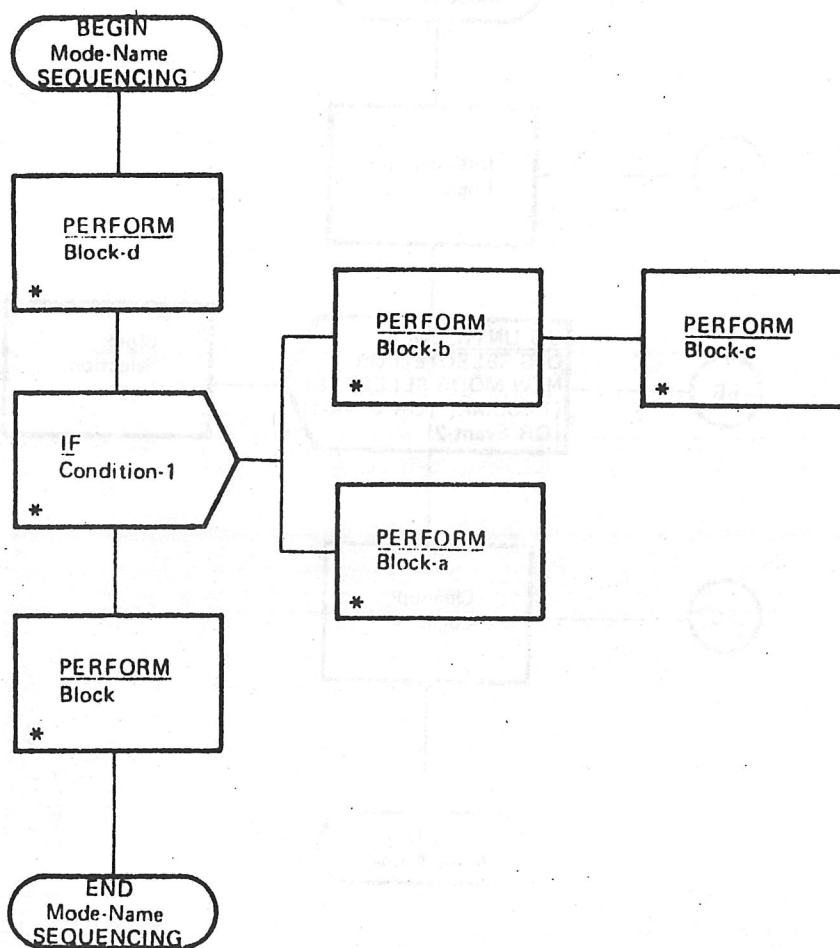


Figure 3.2.3-7 MODE Name SEQUENCING

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

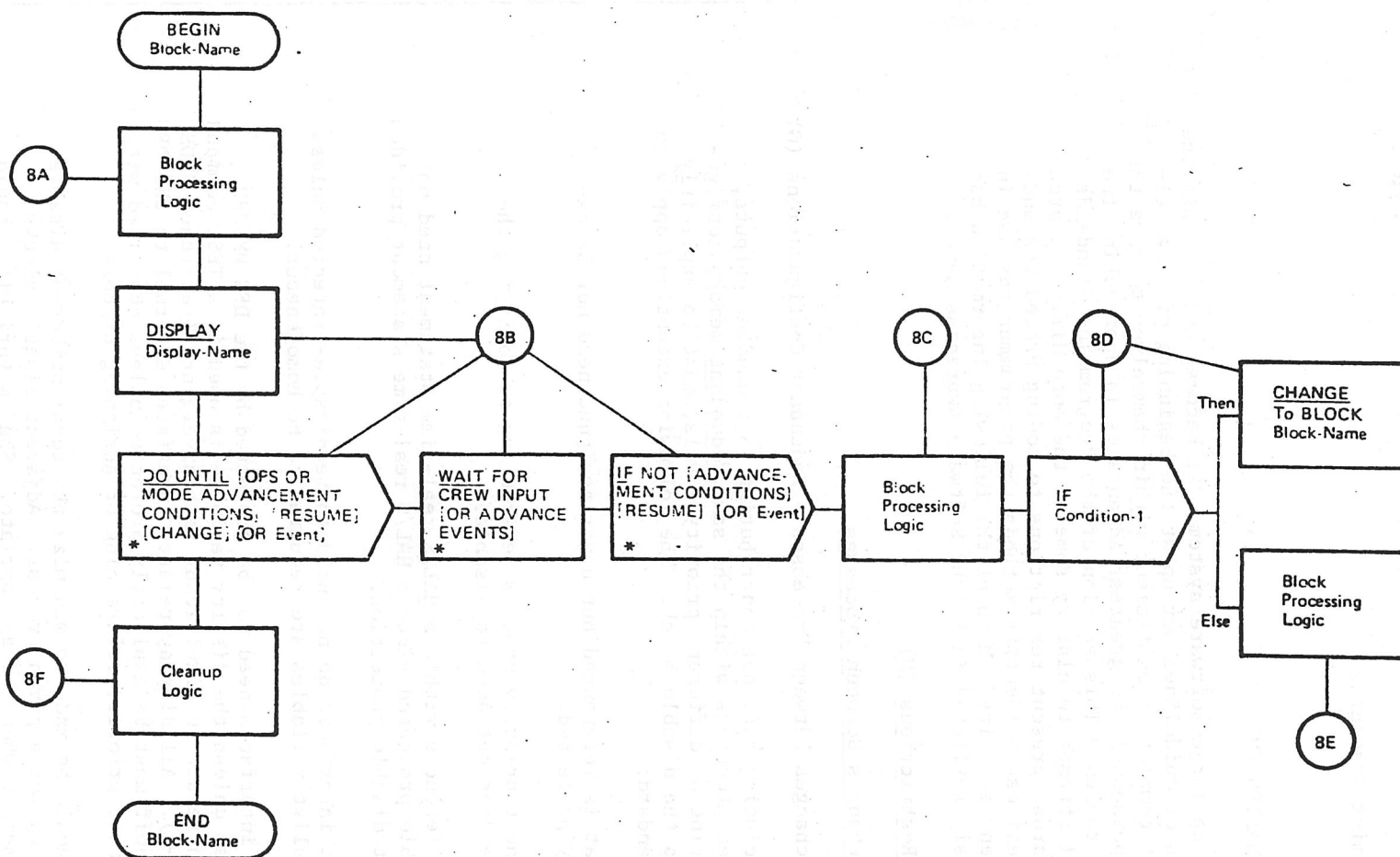


Figure 3.2.3-8 BLOCK Name SEQUENCING

FOR RESEARCH USE ONLY.  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE).

COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY



BOOK: Programming Standards

## 3.3 SOFTWARE ANOMALIES (M)

When developing large software systems, it becomes evident that the general rules or guidelines set up at the beginning of these efforts are not completely sufficient within themselves to give the development personnel the greatest advantages in approaching the development problem. This section of the Programming Standards Document will attempt to clarify some of the more difficult areas of the standards, present restrictions to coding procedures and provide guidelines to identify methods the programmer may use in the development activity. Much of the following information has been previously reflected in Flight Software Awareness Memos.

3.3.1 Coding Restrictions (M)3.3.1.1 Data Exchanges Between Processes

## 3.3.1.1.1 Data Exchanges Between Processes - Redundant Configurations (G)

All data, including data not contributing to redundant outputs, passed between processes within the same redundant memory configuration executing at different priority levels, must be explicitly protected via the disable block. The following exceptions apply to the above standard:

- Data that is referenced but never assigned need not be explicitly protected.
- The highest priority process referencing or modifying the variable does not need to disable.
- Event references within a HAL/S real-time statement need not be disable protected since a HAL/S real-time statement provides implicit disable protection.
- Downlist interfaces do not need to be disable-protected unless the downlist variables are required to be homogeneous.
- Display interfaces need not be protected by the DFG DMDUPD mechanism unless the display variable is used in a TEST command that can result in significant path divergence (reference STDS 3.3.1.5-4). All display-related interfaces external to the DFG code itself must be explicitly protected unless contained within the highest priority referencing or assigning process.

An attempt should be made to minimize the total number of disable blocks required for a given process. Adjacent disable blocks should be combined whenever appropriate. Shadow variables should

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE).  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY

also be used to reduce the total number of disable blocks required. As a goal, cyclic processes should be restricted to no more than two disable blocks - one for inputs and one for outputs. All other communication should be performed by shadow variables.

The practice of using subscripting as a means of IPV protection or partitioning subscripts differently for different processes within the same data aggregate shall be avoided. An attempt should be made to ensure that similar referencing considerations exist for all subscripted elements of a data aggregate (e.g., array elements, structure copies, bits in bit string).

#### 3.1.1.1.2 Data Exchanges Between Processes - All Configurations (G)

All data passed between processes executing concurrently at different priority levels must be protected via the disable block if any of the following conditions exist:

- Multiple variables are explicitly or implicitly (by design) required to be homogenous. The entire set of references to the homogenous set of data shall be enclosed in a common disable block for all processes except the highest priority referencing or assigning process.
- An intermediate assignment of a variable could result in incorrect usage of the data by a higher priority process. The intermediate assignment shall be enclosed within a common disable block with the final assignment by the process to prevent a higher priority interrupt between the two assignments.
- Multiple assignments to the same variable by different processes could result in invalidation of the higher priority assignment. This usually results when the lower priority assignment is determined by referencing other variables that are computed elsewhere within the same process. If an invalidation of the higher priority assignment can occur, the lower priority assignment shall be enclosed within a common disable block with any dependent variables used in the computations of the lower priority value.

#### 3.3.1.2 Restrictions on EVENT Variables in Non-R/T Statements (M)

To ensure that all processes in the Redundant Set (RS) execute with identical data, EVENT variables shall not be tested by any process or used in any bit expression unless it adheres to the same rules as stated in paragraph 3.3.1.1 above. The restricted forms of testing are 1) IF, 2) WHILE, and 3) UNTIL except when they appear on a Realtime Statement (i.e., SCHEDULE and WAIT).

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE

PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)

COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY MANNER NOR PLACED IN ANY REPOSITORY

Examples of restricted forms of bit expressions are:

- 1) BITVAR=EVENTVAR;
- 2) BITVAR=EVENTVAR and BITVAR;
- 3) IF (EVENTVAR) THEN...
- 4) DO WHILE EVENTVAR;
- 5) DO UNTIL EVENTVAR;

Note that these forms do not result in a SVC; therefore, FCOS cannot supply an implicit sync.

### 3.3.1.3 FSW Process Priorities, Rates and Phase Off-Sets (M)

The SARB will control via SAM 10 the assignment of priorities, phases and execution rates to all processes executed in the GPC during Shuttle missions. Any changes to SCHEDULE Statements, related logic, or this baseline are to be requested by submitting FAIRs and reviewed for impact to the I/O profile (SAM 20).

The SARB will control the assignment of priorities to the I/O transactions and the availability of BCE programs in each memory configuration. Changes to this baseline are requested by submitting a FAIR to the SARB after insuring the FAIR'S consistency with the I/O Profile (SAM 20).

Applications programs are to use predefined names when designating process priorities, phasing, and rates. No actual numbers are to be coded in applications programs. Actual numbers will be defined at the system level and are included into the compilation via compiler directive (D INCLUDE ZPRIOTIM). The ZPRIOTIM source library will contain a REPLACE statement for each program name using the standard name conventions with the character string 'PRIO\_ABB' for priorities 'PHASE\_ABB' for phasing, and 'TIME\_ABB' for rates as a prefix to ensure uniqueness. ABB will be equal to the first three characters of the program name. If multiple SCHEDULE statements are defined with different user parameters, a fourth character shall be added to make the REPLACE statement unique (i.e., 'PRIO\_VAA', 'PRIO\_VAA1').

Process dynamics for SCHEDULE statements shall be specified as follows:

```
SCHEDULE      ABB_C...C AT PHASE_ABB PRIORITY (PRIO_ABB),  
              REPEAT EVERY TIME_ABB;
```

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY

#### 3.3.1.4 Redundant GPC Calculations (M)

The GPC redundancy management approach is to assure that the several GPC's comprising the redundant set compute identical results. For the most part this is accomplished by the Systems Software setting COMMFault indicators for applications code.

The Systems Software does require cooperation from the applications programmers in adherence to the following rules:

1. All data (including event variables) passed between processes executing concurrently at different priority levels must be protected via the disable block unless accompanied by HAL/S Real-Time statements (see paragraph 3.3.1.1).
2. Don't use time values derived from the GPC's internal oscillator, since such values are not identical in all computers. The restricted application names are RUNTIME and MET SVC. The restricted FCOS internal names are FPMGMTIM, TCVTSWCH, and TCVTSWCM.
3. Don't make the code dependent upon the timing of external devices. For example, one might execute a processing loop until an I/O operation on Mass Memory was completed. Again this would result in the computers executing the loop a different number of times with subsequent divergence.
4. Don't use GPC discrettes or PCMMU data in redundant set processing without exchange of the data (i.e., ICC) and the execution of a common redundancy management algorithm. This includes, but is not limited to, MMU status discrettes.

There are certain cases where the requirements or hardware design dictate that the GPC's take different processing paths. These will each have to be closely scrutinized to assure that synchronization is not comprised. As each such instance is discovered a FAIR is to be written to the SARB requesting that this violation be examined and documented in SAM 2.

#### 3.3.1.5 Time Constraints on Software Sequences (M)

To ensure reasonably responsive software, the following time constraints are to be adhered to:

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY

1. The maximum execution time within an EXCLUSIVE procedure will be constrained to be less than two milliseconds. Application disable periods should be less than 550 microseconds. These constraints are intended to assure that higher priority processes are not overly impacted by sharing data and programs with lower priority work. This timing constraint also assumes approximately 250 microseconds disable time in FCOS. Application disable periods greater than 550 microseconds shall be reviewed by the SARB for possible violation of Level A jitter requirements. Application disable periods shall not exceed 550 microseconds in the redundant set and 3.0 milliseconds in the common set (non-redundant), in order to prevent combinations of disable periods and GPC-unique effects (e.g., unique I/O errors, memory interference) from approaching the sync time-out tolerance.
2. No application process, except control segments, shall execute for more than 1/2 second without executing a CLOSE or checking for a termination request. Note that a continuation of processing can be attained by cyclic reactivations of the process. This constraint is intended to assure that the process CANCEL can be effected to allow sequencing such as SPEC deactivation and OPS transitions.
3. Control segments shall not execute for more than 25 milliseconds without presenting a display and accepting keyboard inputs. This constraint is to ensure reasonable response to keyboard inputs. The only exception is the DEU Self-test SPEC where the control is relinquished by Flight Software.
4. For software which can execute in a redundant set, the sum of the differences in timing on all different paths which can be taken based on unprotected data between two sync points must be used to determine whether the process can miss sync. In order to ensure not exceeding the sync tolerance, the design goal for the timing differences between paths leading to a common execution point is less than 500 microseconds - composite path skew for a process should not exceed 1500 microseconds. This time difference permits other non-universal effects (such as I/O error processing in only some of the computers, and memory contention by the CPU and IOP) from reaching the sync time-out tolerance by an accumulation of such effects.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY MANNER NOR PLACED IN ANY REPOSITORY

## BOOK: Programming Standards

5. For software which can execute in a redundant set, no process shall execute more than 100 milliseconds between statements that issue an SVC with sync. This value assumes identical processing paths between the sync points. With the maximum processing difference of 1.5 msec (above), the interval between sync points is reduced to 60 msec. For processing differences between 0 and 1.5 msec, a linear interpolation from 100 to 60 msec provides the maximum interval between sync points.

The 60 to 100 msec interval is based on a slow-down in CPU processing speed caused by contention for memory by the CPU and the IOP. In particular, when most data buses are commanded by one GPC, that command GPC has many more I/O instruction and data memory accesses for outputs than do the non-commanding GPCs (which make no memory accesses for output I/O). The specified interval permits the unbalanced I/O to occur during any process without the cumulative effect approaching the sync time-out tolerance.

6. To accommodate natural variation of normal processing and error conditions, the HFE output data homogeneity margin shall be 500 microseconds, i.e., flight critical processing shall complete at least 500 microseconds prior to the point in the HFE output transaction which sends the computed data to the MDMs.

The above times are not to consider interruptions by FCOS or higher priority work.

Exceptions are to be requested by submitting FAIRs to the SARB.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY



### 3.3.1.6 Restricted Use of HAL/S Real-Time Features (M)

Since ON ERROR IGNORE may be used to "mask" or bypass software and/or hardware failures, any use must be approved by the SARB. Any use of the TERMINATE statement must also be approved by the SARB.

### 3.3.1.7 Characteristics of HAL/S SCHEDULE Statements (M)

FCOS is designed such that when the SCHEDULE statement is used with both the AT and REPEAT EVERY option, special phasing occurs. That is, when the AT time is less than current time, the AT is projected by integral repeat interval times to determine the time of initial execution.

It should be recognized that when the initial execution is keyed upon an event expression that this phasing does not occur. Indiscriminate use of the EVENT scheduling HAL/S option or default (no AT expression) could introduce jitter into other processes.

If process phasing with other processes is desired, do not use event expressions and REPEAT EVERY and assume this happens automatically. Phasing can be accomplished by activating another process upon event occurrence which in turn executes a cyclic SCHEDULE using the AT and REPEAT EVERY time options.

### 3.3.1.8 Assembly Language Usage (M)

Each new use of assembly language is to be reviewed by the SARB (via an FAIR form) prior to NASA presentation. This procedure is being established to have a coordinated FSW approach in the use of assembly language, and to ensure alternative workaround solutions have been examined. The existing approved exceptions are FCOS and Cyclic Display Updating.

When assembly language is to be used, the development groups will employ the structured programming macros. Exceptions may be requested by submitting a FSW Action Item Request (FAIR) to the SARB.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY

### 3.3.1.9 Constraints When Writing Exclusive Disable/Procedures Blocks (M/G)

#### 3.3.1.9.1 Constraints When Writing Disable Blocks (M)

Disable blocks define regions of code during the execution of which neither timer nor I/O interrupts should be accepted. The purpose of the disable block can be defeated in subtle ways by most HAL real-time statements and macros which invoke FCOS services. Because the effects are not obvious and can be difficult to analyze, and because alternate program constructs are usually possible, the following types of statements should not be used within disable blocks (all necessary uses are to be documented in SAM 2 after approval by the SARB):

1. All HAL real-time statements
2. All macros which invoke FCOS services
3. External procedure or function invocations.

#### 3.3.1.9.2 Constraints When Writing Exclusive Procedures (G)

Exclusive procedures are dedicated to one process until the CLOSE of the procedure. They, therefore, allow one process to block execution of higher priority processes which share that procedure. The users of exclusive procedures must assure that such delays of higher priority processes are acceptable. The following statements contain potentially hidden delays which must be considered in addition to normal processing.

1. Explicit WAITs and WAITs embedded in input/output
2. Any statement which changes an EVENT that may initiate a higher priority process (i.e., SET, RESET, SIGNAL, SCHEDULE, CANCEL, TERMINATE)
3. SCHEDULE of a higher priority process
4. CALLS to other procedures (internal or external) which contain any of these statements.

Exclusive procedures are not to be shared with flight control, since a time-out at the related sync point could violate flight critical I/O data homogeneity requirements.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FORM NOR PLACED IN ANY REPOSITORY



### 3.3.1.10 Use of ON ERROR for I/O Operations (M)

The use of ON ERROR is controlled by the SARB for flight software. In the event such approval is requested, then the organization described in the following paragraphs must be followed.

The ON ERROR should always be issued before and maintained during any I/O operations, associated with the process, that could set the specified error. Otherwise the ON ERROR environment may not be established when the actual error conditions are met. This is not desirable in a simplex system because of the increased complexity and restrictions on I/O timing. In a redundant set, different sequences could be taken and sync points missed or reordered in some members of the set thus breaking up the redundant set.

Specifically, the ON ERROR statement should be placed before the I/O request statement. The WAIT option should be used on I/O or an explicit WAIT coded prior to another ON ERROR statement with the same error number. The WAIT should also be used prior to any RETURN or CLOSE statement that completes processing of the program or procedure that issued the ON ERROR.

The ON ERROR statement establishes an error processing environment that is inhibited by a subsequent ON ERROR for the same error. The original environment or override environment is automatically deleted when processing is completed for the level that contains the corresponding ON ERROR. Again, a WAIT must be placed prior to any statement that can change the error environment.

### 3.3.1.11 Protected I/O Transactions (M)

The concept of protected I/O transactions allows the status of I/O operations (specifically inputs) to be determined in all GPC's of a redundant set via an I/O parameter list specification.

Protected transactions are necessary because GPC's in the redundant set must receive bit-for-bit identical input data. Since this is a requirement for our redundancy management approach, the same safeguards as for nonidentical processing are to be employed. Specifically, all input transactions are to be "protected" unless all processes using the data have been identified and accepted as non-identical computations (e.g., PCMMU).

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

## BOOK: Programming Standards

3.3.1.12 Interprocess Variables Not Protected by Disable Blocks (M)

In general, paragraphs 3.3.1.1.1, 3.3.1.1.2 require explicit disable protection for IPVs. In cases where exceptions are permitted, adherence to several restrictions on the usage of unprotected data is required. These restrictions can be divided into two categories: those which apply only to programs which must run in a redundant set and those which apply to all programs.

- a. Restrictions which apply only to programs which must run in a redundant set.
  - 1) Do not test unprotected variables to decide whether to take a path resulting in a sync point. All I/O and many SVCs, such as UPDATE, SCHEDULE, CANCEL, SET, WAIT result in a sync in a redundant set.
  - 2) Do not make tests on unprotected data where the paths to be taken can differ in CPU utilization enough to miss the next sync point in the process. Since the close of a process is a sync point, there is always a next sync point in the process. The sum of the differences in timing on all different paths which can be taken based on unprotected data between two sync points must be used to determine whether the process can miss sync. In order to ensure not exceeding the sync tolerance, the design goal for the timing differences between paths is less than 500 microseconds.
  - 3) Do not use unprotected data to set flags or compute data on which other program segments make decisions which violate restriction 1 or 2. If unprotected data is used to compute other data, the computed data is also unprotected, even if it is stored and read under update protection. This restriction means that any areas which use data received from display processors, SM, or downlist must observe restrictions 1 and 2 in their use of that data if they reside in the redundant set.
- b. Restrictions which apply to all programs.
  - 1) Interprocess data which must be time homogenous must be disable protected. Except for non-homogenous data for downlist and display, do not change the contents of an output buffer while output is in progress.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY

## BOOK: Programming Standards

- 2) A word packed with bits or bit strings by different processes must not be picked up, updated, and replaced by one process without disable protection in every process, except the highest, updating the word. Otherwise, one process could destroy another's update of other bits in the word. Either the bits must be updated without removing the word from its location in storage or a disable block must be placed around the whole operation (from before the word is read until after it is stored). The following are examples of HAL statements which may be expanded by the compiler into a load, bit manipulation, and a store.

(a)  $A_i = B_j;$       (b)  $A_i, A_j = \text{OFF};$

where A and B are bit strings and  $A_i$  and  $A_j$  are contained in the same word.

- 3) Unprotected interprocess data should not contain intermediate values not truly reflective of the variable. This requirement is mandatory except for data interfacing with displays and/or downlist. Prohibition of intermediate assign interfaces will be mandatory for display/downlist interfaces upon approval of an appropriate authorization change request. It is recommended, however, that intermediate values associated with these interfaces be avoided whenever possible.

Example sequences that illustrate the problem are as follows:

(a)  $Y = A+B;$   
 $Y = Y+C;$

(b)  $A = 0;$   
 IF B THEN  
 $A = 1;$

In the first example, Y takes on an intermediate value that may be misleading if sampled between the first and second statements. Between the first and second statement of the second example, A contains a value not indicative of state B. If downlist, SM, or display update captured the value of A between the statements, confusion could result.

FOR RESEARCH USE ONLY  
 THIS MATERIAL MAY BE  
 PROTECTED BY COPYRIGHT LAW  
 (TITLE 17 U.S. CODE).  
 COPY PROVIDED BY  
 WICHITA STATE UNIVERSITY LIBRARIES  
 SPECIAL COLLECTIONS  
 WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
 REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

BOOK: Programming Standards

The proper way to code such sequences are:

(a)  $Y = A+B+C;$

(b) IF B THEN A = 1;  
ELSE A = 0;

Similarly, avoid using such variables as temporary storage or to contain intermediate results.

Note that this restriction applies to any processes storing unprotected variables rather than to the programs accessing the data.

#### 3.3.1.13 Processes Executing Over OPS Transitions (M)

System Software cancels all application processes if a transition involves adding a GPC to the Redundant set that was not previously in the set executing the function. Function Base data is transferred to the new GPC but the application is responsible for re-scheduling the processes that were to execute across the transition and for the initialization of other redundant data.

#### 3.3.1.14 Checksums (M)

Checksums shall be generated and written to mass memory for each individually selected set of data or code. Each read of these sets will be followed by a test for correct checksum if a loss of data can affect performance.

#### 3.3.1.15 RIGID COMPOOLS/STRUCTURES (M)

All COMPOOLS/STRUCTURES shall be defined with the RIGID attribute. If not specified, the compiler will determine data allocation based on its optimization algorithm. This lack of predictability could result in a code generation problem in cases where code sequences assume a specific data ordering (ex., use of %COPY statements).

#### 3.3.1.16 Local Data %COPY (M)

%COPY operations involving local data shall not be used unless the copy range is totally contained within a single local variable. This is to preclude unpredictable effects due to compiler data allocation optimization.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION. THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY.

## BOOK: Programming Standards

3.3.1.17 FSW Error Protection (M)

Ensure that the software implementation will preclude the possibility of software error conditions (i.e., program checks, instruction monitor, fail to sync, GPC error log, out of range case, etc.). To evaluate the need for protection, consider the potential risk due to future changes, not just the current need for protection. Unless memory/CPU resource costs are prohibitive, provide appropriate (as defined in SAM 28) error prevention protection even if it is not currently required:

- If any reasonable possibility exists that future changes to external constraints (ex., ILOADS, crew inputs, limit checks, other external code changes) could go undetected.
- If it will be difficult or time-consuming to establish that no error prevention protection is needed even with well-documented source code, other documentation.

If memory or CPU costs are prohibitive, commenting sufficient to preclude problems from future changes should be added to all related code for which future changes could cause a problem.

3.3.2 Coding Guidelines (G)3.3.2.1 Removal of Unnecessary Diagnostic Messages (G)

Diagnostic messages generated by the language processors and the linkage editor are intended to indicate a problem or potential problem that requires review/correction by the programmer. When diagnostic messages become standard or 'expected', actual problems are often not noticed.

It is required that assemblies/compilation of all software contained on the FSW master system produce no diagnostic messages. Furthermore, load modules on the master system must be produced by the linkage editor with messages that can be readily reviewed.

3.3.2.2 Source Macro Definition Considerations (G)

The maximum total size for all Replace statement text in a HAL/S compilation is limited only by the region size in the current compiler. Currently a significant amount of space is lost through the following coding practices:

1. Excessive or extraneous blanks (e.g., multiple card input).
2. Embedded comment fields (most INCLUDEs are NO-LIST).

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION. THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

BOOK: Programming Standards

3. Non-functional grouping (resulting in many unreferenced symbols).
4. Use of Numeric Replace rather than declared constants.
5. Long names for parametric and global symbols (e.g., use more "ç" catenations).
6. Non-essential use of replace symbols, replace levels, or grouping.

Before new macro statement source members are defined, examine the overall impact to programs that are expected to INCLUDE the member. When updating existing members consider reduction in one or more of the listed areas.

### 3.3.2.3 FSW Data Referencing Considerations (G)

#### 3.3.2.3.1 Optimization Considerations (G)

Three programming practices result in unreferenced FSW code or data residing in most memory configurations. The capacity of the AP101 memory is 106,496 fullwords (32 bit words) and, very often, we are approaching this limit in several configurations. The three practices are explained below. Steps should be taken to avoid occurrences in the flight software.

1. Unused Template INCLUDE Statements - a Template INCLUDE statement is adequate by itself to cause the linkage editor to AUTO-CALL the referenced compilation unit. This is independent of whether or not the included compilation unit is referenced otherwise by the including compilation unit. The result is that COMPOOLS, or other compilation units, are AUTO-CALLED into memory configurations for which they are never used resulting in a memory penalty.

For example - If module A contains an INCLUDE statement for module B, B will always reside in the same memory configuration as A regardless of whether or not A references B. To eliminate memory penalties associated with this situation, unused template INCLUDE statements should be avoided. Otherwise, a manual effort is required during each build to locate such entries and to override AUTO-CALL.

2. Improper COMPOOL Utilization - a single reference to a COMPOOL data item is adequate to effect placement of the entire COMPOOL in the memory configuration associated with the reference. In many cases, the memory penalty is significant. To avoid this penalty, care should be exercised in placement of COMPOOL variables to ensure that most of the data contained in the

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY



COMPOOL is actually used in the memory configuration(s) containing the COMPOOL. For example - suppose COMPOOL CGZR13 is initially intended to contain only data for OPS1 and OPS3. If an OPS2 module references a piece of data (say parameter C) in CGZR13, the whole COMPOOL is brought into the OPS2 configuration by the linkage editor. The correct procedure is for the programmer to define the desired piece of data (parameter C) in another COMPOOL that is intended for OPS2 usage.

3. Unreferenced COMPOOL Data - there is a significant number of declared data variables that are either never referenced in any memory configuration (totally unreferenced) or are referenced in only some of the memory configurations in which they reside (selectively unreferenced). This is primarily due to development fallout when requirements or design changes eliminate the need for referencing certain variables.

Caution: HAL/S name scoping rules will result in the COMPOOL variable not being referenced by a module that redefines or uses the same name as a COMPOOL variable.

Totally Unreferenced Data - COMPOOL data that is never referenced in any memory configuration. Unless used for buffer pad, totally unreferenced data should be avoided. This data should be identified at the time the data becomes unreferenced. It should either be removed from the COMPOOL or the data name changed to include an appropriate identifier (i.e., UNUSED) to designate the data item as a future scrub candidate.

Selectively Unreferenced - COMPOOL data which exist in more than one memory configuration but is unreferenced in some but not all of those configurations. For example - COMPOOL CGZ123 contains parameters D, E, F and is contained in OPS1, 2, 3. Parameters D and E are referenced in OPS1, 2, 3, but F is only referenced in OPS 1 and 2. Selectively unreferenced data is caused by improper COMPOOL utilization practices described in (2) above and such practices should be avoided.

Note that FSW analysis utilities can be used to assist in assessing optimization potential due to improper COMPOOL data referencing considerations. Utility, TMPL, will provide a listing of all unused Template INCLUDE statements. Utility, CMPL, will provide a detailed summary of totally unreferenced and selectively unreferenced data.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY

### 3.3.2.3.2 Reliability Considerations (G)

CONCARDS updates are required to suppress the autocall of data or code CSECTS that are referenced by a compilation unit but are precluded by the design from being used within a memory configuration. A FSW problem could result if future design changes eliminate the referencing protection. The following guidelines are intended to minimize such exposures.

1. COMPOOL data referenced by code outside of FCOS should be memory resident even if the data is precluded by the design from being used within a memory configuration. In general this will eliminate the need for CONCARDS to suppress COMPOOL AUTOCALLS.
2. For code blocks referenced but not contained within the same memory configuration, it should be obvious from the context of the code block invocation (CALL, SCHEDULE) how the invocation is protected from execution. If an explicit memory configuration check is not defined, appropriate commenting should be defined to indicate the protection rationale.

Appropriate CONCARDS updates (change/include, not LIBRARY\*) will be made to ensure any reference to a non-existing code block will result in an error message being logged.

Compilation unit structuring should be used that allows adherence to the above standards. Code segments that have different memory configuration residency requirements than the invoking compilation unit should be partitioned into separate compilation units. This will permit adherence to (1) above. CONCARDS may then be used to define a different memory configuration residency for the invoked compilation unit as long as compliance with (2) above exists.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)

COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY



#### 3.3.2.4 Restricted HAL Statements (G)

HAL/S constructs that result in hidden XREF information that cannot be resolved automatically will be avoided unless it can be demonstrated that their usage clearly results in significant efficiency or other improvements over an alternative approach. This will reduce analysis costs required to compensate for interface deficiencies. FSW reliability will be enhanced accordingly. The following are HAL/S constructs to be avoided:

- %NAMEADD
- %NAMECOPY
- %COPY with variable count
- CARDTYPE
- Integers as address pointers
- Nested structures
- NAME operands in %COPY
- EQUATE HAL name same as NONHAL name

Usage of all %MACROS should be avoided when reasonable alternative language constructs exist. Not only may hidden XREFs result, but other undesirable effects may be created. For example, manual procedures are required for the spill compiler to properly process COMPOOLS containing data referenced by %COPY statements.

In addition, the following restrictions apply to HAL/S INCLUDE statements:

- Included code segments will be listed (i.e., NOLIST will not be specified).
- Macro include segments will be defined within a COMPOOL rather than locally.
- Except for HAL-compatible interfaces to FCOS modules defined in the FCOS User's Guide, non-compiler generated templates will not be used (i.e., except as noted, the EXTERNAL attribute should not be used on any block header statement).

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY MANNER NOR PLACED IN ANY REPOSITORY

## BOOK: Programming Standards

## 3.3.2.5. Data Initialization (G)

All HAL/S DECLARE statements except for stack variables (Temporary, Automatic, calling arguments) will have an appropriate initial value defined with the INITIAL or CONSTANT attribute. This standard is primarily intended to ensure that predictable values result that are not effected by changes beyond the programmer's control (e.g., compiler, linkage editor, mass memory build).

The programmer should also ensure that other initialization techniques are used as appropriate. The programmer should not rely on DECLARE initialization as the only method of initialization unless the OPS will always be initiated by an overlay from mass memory. The only OPS satisfying this constraint is G1/6.

For other initialization considerations, reference SAM 26 (Ensuring Data Integrity at OPS Transitions or OPS Mode Recall) and the initialization design/code inspection checklist item description in SAM 31.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY

WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS

WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

BOOK: Programming Standards

## 4. DOCUMENTATION

This section will discuss documentation standards and requirements that are applicable to flight software. Two types are considered here. Documentation in the listing shall be used to assure readability of the flight software code. Mandatory (M) and Guideline (G) paragraphs will be marked.

Control documentation through the usage of pre-defined forms will serve to communicate information concerning flight software changes, problems, updates, etc., to the appropriate sources.

## 4.1 SOURCE LISTINGS (M)

The listing of flight software source code is generated by the output writer of the HAL/S compiler and the AP-101 assembler. Although these listings are largely self-documenting, each module will be prefaced with a set of summary comments. Also, the source code will be annotated throughout with comment statements and/or inline comments.

4.1.1 Prologue Comments (M)

Prologue comments will be included within every programmer generated or preprocessor generated FSW source member. This includes programmer created preprocessor input source members. In general, the prologue comments will appear at the beginning of the source member immediately after the header statements. To prevent source code resequencing, it is acceptable to provide a reference to the update history provided at some other point in the source code within that member (i.e., end of member). Prologue comments will comply with the following format:

- |                     |   |  |
|---------------------|---|--|
| MODULE NAME         | - | HAL or NONHAL name of the code block<br>(e.g., GEA_ASC_RTLS_HFE)                               |
| DESCRIPTIVE<br>NAME | - | Brief English title of the code block<br>(e.g., Ascent/RTLS High Frequency<br>Executive)       |
| PURPOSE             | - | Brief summary of purpose of the module   |
| CHANGE<br>ACTIVITY  | - | Update history of all authorized changes<br>made to the program to include the fol-<br>lowing: |
|                     | - | Authorization ID-a CR, PCR, or DR<br>number is required for all programmer                     |

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY INFORMATION

## BOOK: Programming Standards

created changes. Alternate authorizations (e.g., ICD number) may be specified for preprocessor created source members if change traceability is enhanced.

- Date of change
- Programmer identification (preferably programmer initials)
- Title or description of change
- Two-character revision level associated with the update (should correlate with the PMF-generated revision level for the update)

Optional information, such as the build to which the change is applied, other modules changed for this authorization, ID, related authorization IDs, etc. may be included.

Other prologue comments such as the following may also be included but are not required:

- |                        |   |  |
|------------------------|---|--|
| INPUTS                 | - | Name and description of each parameter externally provided to the module as a CALL argument.   |
| OUTPUT                 | - | Names and description of each CALL argument that may be altered by the module even if they are also inputs (COMPOOL parameters are included but local data is excluded). |
| NESTED<br>CODE BLOCKS  | - | Description of each code block internal to this module including name and type.  |
| EXTERNAL<br>REFERENCES | - | External modules invoked by this module.   |
| ERROR<br>CHECKS        | - | Brief description of any error conditions checking performed, and associated return codes.   |

FOR RESEARCH USE ONLY  
 THIS MATERIAL MAY BE  
 PROTECTED BY COPYRIGHT LAW  
 (TITLE 17 U.S. CODE)  
 COPY PROVIDED BY  
 WICHITA STATE UNIVERSITY LIBRARIES  
 SPECIAL COLLECTIONS  
 WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
 REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

#### 4.1.2 Authorization ID Correlation Comments (M)

Information must be included in each source member update to precisely correlate each source statement changed, added, or deleted to all applicable change authorizations. The method used to accomplish this shall not significantly decrease the readability of the program code and comments.

If a single change authorization provides the basis for all changes to a member for a revision level and no statements are deleted, the update history entry for the change will provide adequate correlation via the source line revision level information. In other cases, additional comments must be used to provide the necessary information. Comments may be included with the change history to indicate Statement Reference Numbers (SRNs) or SRN ranges associated with the change or comments may be included on or near the statements changed. If any statements were deleted, comments must indicate their SRN's and specify that they were deleted.

Enforcement of this Standard shall be provided by the Code Inspection process. In order for a code change to be considered complete, the update history must be properly updated and all source statements changed must be properly correlated to all applicable change authorizations. In the case of overlapping changes or updates to statements previously changed, the authorization ID's for all changes must be identified in an unambiguous and easily understood manner. Comments included in source members in order to comply with this Standard are considered part of the source member source and therefore may be changed or deleted only with proper authorization (CR, PCR or DR). Any authorization to change a member may be used as a basis for (1) minor changes, additions, or deletions to comments describing prior code, (2) minor program label updates, (3) deleting unnecessary template include directives, or (4) similar changes to compiler/assembler directives (e.g., non HAL TITLE CARDS, EJECT, SPACE) that do not produce object code provided that they are reviewed as part of the code inspection and that the resulting source program comments comply with this Standard. Change authorization ID comments should identify the change authorization for which the source member was opened as authorizing the comment/label/directive changes. If any comment/label/directive changes are made beyond the scope of the authorizing document, the DR closure or PCA (for CRs and PCRs) must reference those comment/label/directive changes.

This standard is not applicable to source members wholly created by a preprocessor. Compliance is required, however, for any manual updates to preprocessor created source members as well as for programmer created preprocessor input source members.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION, NOR PLACED IN ANY REPOSITORY

BOOK: Programming Standards

#### 4.1.3 Statement Comments (G)

The HAL/S compiler generates a great deal of self-documenting source code. However, the programmer must supply comment statements and/or inline comments (i.e., on applicable source code line itself), wherever the purpose or effect of a source statement or set of statements may not be self-documenting. Comments must be provided for the following situations:

- o When efficiency requirements preclude straightforward coding, comments must be included to clarify the function of such code. Assembly language code shall include sufficient comments to place it on a par with HAL/S code with respect to readability.
- o Instructions that may cause branching must have comments to explain the branch test conditions and the significance or nature of each possible branch, if not obvious.
- o Statements invoking a procedure or function must be accompanied by comments stating the reason for invocation unless the procedure/function name identifies its purpose.

Reference SAM 15 for HAL/S output writer considerations that influence the placement of comments.

#### 4.1.4 FSW Source Resequencing (M)

FSW source members, except those created by a preprocessor, will not be resequenced unless significant changes are required for the source member. Resequencing of an entire source member must also be approved by the OBS FSW development manager (i.e., third line manager). If whole or partial resequencing is necessary, the authorization closure associated with the updates (DR or PCA) will reflect that resequencing was performed. In addition, the authorization comments (required by Standards paragraph 4.1.2) will properly reflect the fact that a resequencing has occurred. Explicit references to Statement Reference Numbers (SRNs) in the module change history will be updated as appropriate to maintain the integrity of the change history.

This standard is defined due to the high utilization of Statement Reference Numbers (SRNs) in test decks and FSW analysis tasks. Also, SRNs are used as a key in determining FSW source and XREF deltas between systems.

Compliance with this standard shall be enforced by the code inspection team for both partially and wholly resequenced source

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE)  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY FASHION NOR PLACED IN ANY REPOSITORY

BOOK: Programming Standards

members. No SARB review is required. The code inspection team will establish the following:

- Resequencing is necessary or highly desirable
- Appropriate management authorization has been obtained if any entire source member is resequenced
- Impacts to existing IPV, Statement Level Data Base, and other analysis inputs have been properly considered and appropriate action taken
- Authorization ID correlation comments properly reflect the fact that a resequencing has occurred and explicit SRN references are correct

#### 4.2 DESIGN SPECIFICATION FORMAT (M)

The format for FSW design documentation will be as described in SAM 13.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE).  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY MANNER, NOR PLACED IN ANY REPOSITORY

## 5. REFERENCE MATERIAL

- o Interface Control Document: HAL/SDL (IBM No. SS-81-4465) - Rev. 7 defines interfaces specific to HAL/S and SDL software systems.
- o HAL/S-360 Compiler System Specification (Intermetrics Incorporated) - specifies the informational interfaces within the HAL/S-360 compiler, and between the compiler and the external environment.
- o HAL/S Language Specification (Intermetrics Incorporated) - format description of the HAL/S language.
- o Structured Programming (H. D. Mills, 1970) - formal description of structured programming.
- o 'Chief Programmer Team Management of Production Programming' by F. T. Baker (IBM Systems Journal, Volume XI, No. 1, 1972, pp. 56-73) - includes discussion of chief programmer, program production librarian, top-down programming, and structured programming.
- o 'Chief Programmer Teams' by F. Terry Baker and Harlan D. Mills (Datamation, Volume 19, No. 12, December 1973, pp. 58-61) - brief discussion of chief programmer, top-down development, structured programming, and development support library.

FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE).  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS  
WITHOUT WRITTEN PERMISSION, THIS MATERIAL MAY NOT BE COPIED OR  
REPRODUCED IN ANY MANNER FOR ANY PURPOSE.



FOR RESEARCH USE ONLY  
THIS MATERIAL MAY BE  
PROTECTED BY COPYRIGHT LAW  
(TITLE 17 U.S. CODE).  
COPY PROVIDED BY  
WICHITA STATE UNIVERSITY LIBRARIES  
SPECIAL COLLECTIONS