


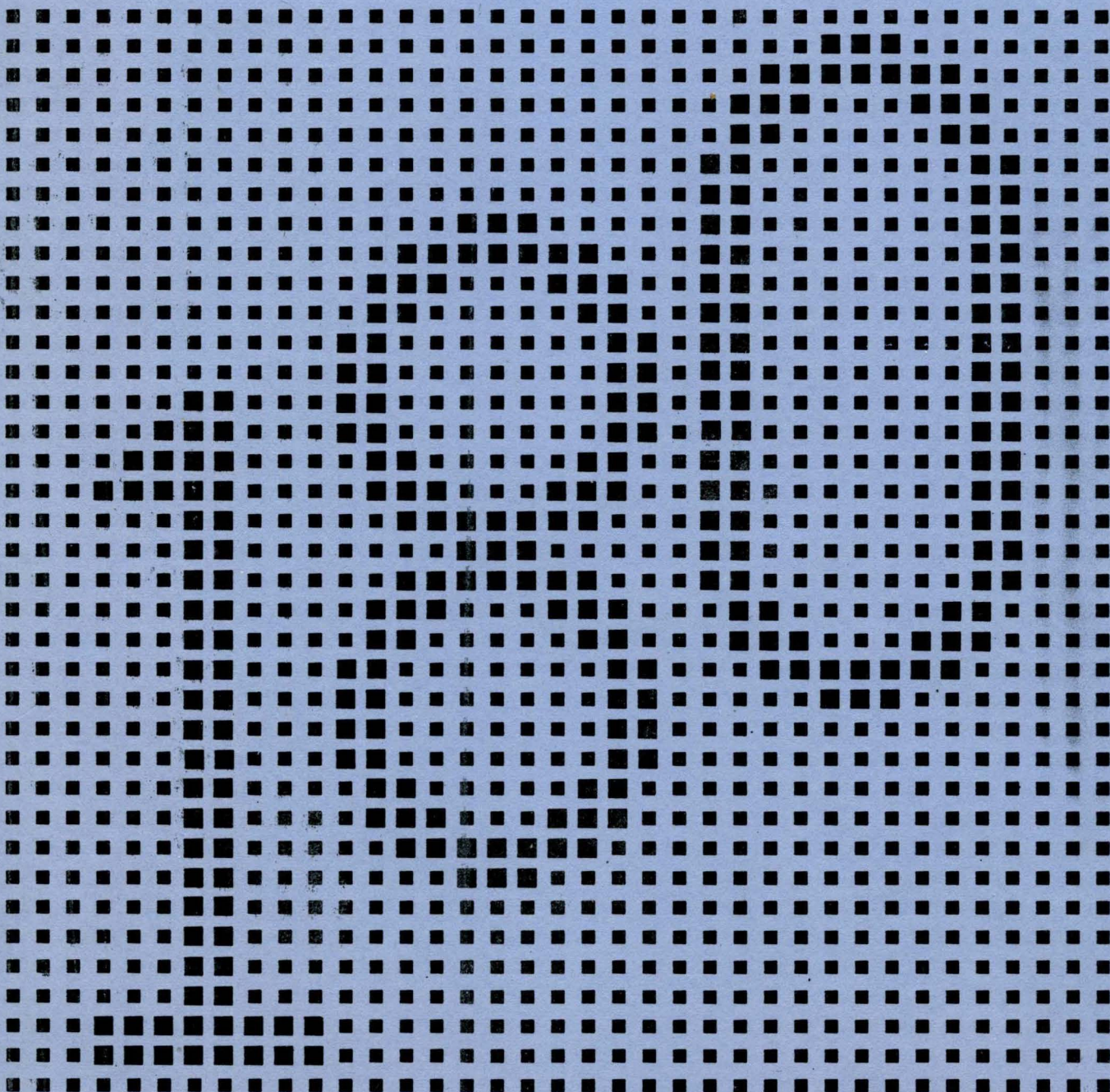
CONTROL DATA  
EDUCATION COMPANY

 a service of  
CONTROL DATA CORPORATION

# CYBER 180 Programming Guidelines

A course in the  
CYBER 180 curriculum

CONTROL  
DATA  
PRIVATE



Student Handout

# CYBER 180 Programming Guidelines

A course in the  
CYBER 180 curriculum

CONTROL DATA  
EDUCATION COMPANY

 a service of  
CONTROL DATA CORPORATION



# CONTENTS

---

Introduction to the Course	iii
I. Introduction	1
II. System Interface Standard	7
III. Performance	25
IV. Project Conventions	33
Appendix	43

---

CONTROL DATA PRIVATE

# **GENERAL COURSE DESCRIPTION**

## **COURSE TITLE**

**Programming Guidelines**

## **COURSE NUMBER**

**RW 3500**

## **COURSE LENGTH**

**One day (5-6 hours)**

## **GOAL**

**Contribute to the production of quality C180 software by increasing awareness of the standards, guidelines, and techniques used by C180 software developers.**

## **DESCRIPTION**

**To attain this goal, the class will work with the System Interface Standard (SIS) and other documents that describe the C180 software development process. The class will participate in discussions, evaluate code examples, and attend some short presentations.**

CONTROL DATA PRIVATE

# **GENERAL COURSE DESCRIPTION (continued)**

## **PREREQUISITES**

The minimum requirements for the course are the asterisked courses listed below. The others are recommended.

- \* C180 Introduction or C180 Hardware
- \* CYBIL
- \* Structured Analysis/Structured Design  
Utility Smorgasbord  
NOS/VE Internals and NOS/VE Usage  
SYMPL

## **COURSE MATERIAL**

1. System Interface Standard (SIS)
2. NOS/VE Project Procedures and Conventions
3. Handout

CONTROL DATA PRIVATE

# OUTLINE

- I. INTRODUCTION (1 hour)
  - A. GOALS
  - B. GENERAL RULES
  
- II. SYSTEM INTERFACE STANDARD (2 hours)
  - A. PRIORITIES
  - B. INPUT
    - 1. Product Calls
    - 2. Source Input
    - 3. Product Directives
  - C. OUTPUT
    - 1. Logs
    - 2. Listable Output
  - D. SYSTEM-WIDE CONVENTIONS
    - 1. Naming
    - 2. Interactive Processing
    - 3. Error Processing
  - E. COMPILER AND ASSEMBLER OBJECT CODE
    - 1. Interlanguage Calling Sequences
    - 2. Support Modules
  
- III. PERFORMANCE (1 hour)
  - A. PROCESS
  - B. LOCALITY OF REFERENCE
    - 1. Code
    - 2. Data
  
- IV. PROJECT CONVENTIONS (1 hour)
  - A. HIERARCHY
  - B. PROJECT CONVENTION DOCUMENT
  - C. CYBIL CONVENTIONS
    - 1. Readability and Clarity
    - 2. Reliability and Safety
    - 3. Performance
  - D. PACKAGING

APPENDIX

CONTROL DATA PRIVATE

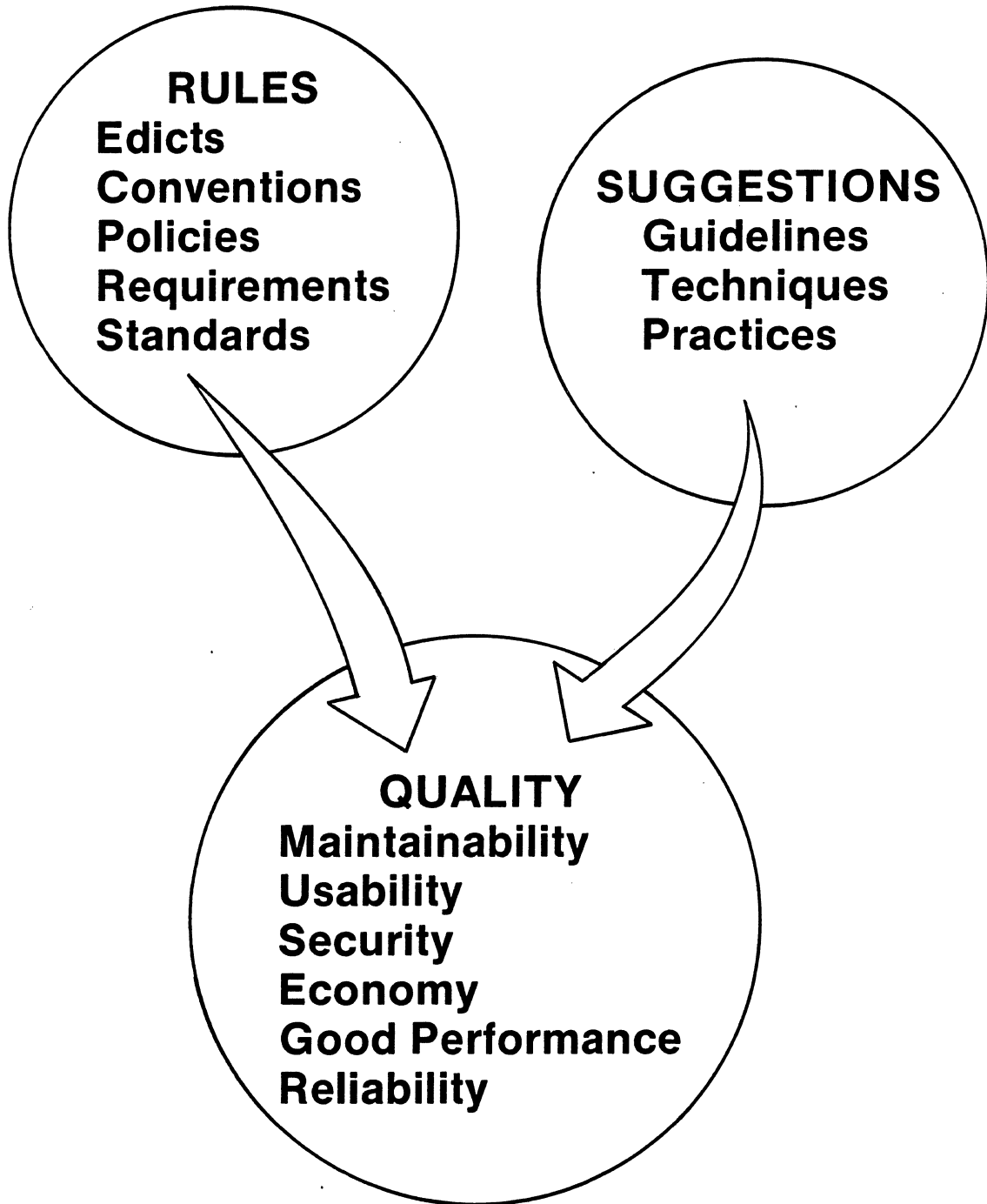
# I

# INTRODUCTION

CONTROL DATA PRIVATE



# GOALS



CONTROL DATA PRIVATE

# GENERAL RULES

- **Use structured methods for all applicable phases of analysis, design, implementation, and test.**
- **Follow the system interface standard (SIS) in all OS/product, product/product, and product/user communications.**
- **Code for simplicity and clarity; measure, then revise.**
- **Follow the coding conventions (for naming, documenting, and so on) listed in the SIS or documents for your product.**
- **Access tools for maintaining, manipulating, documenting, compiling, debugging, and so on through SES procedures.**
- **Use the program interface for message processing, access methods, interlocking, and so on.**

# PROCESS

## **ANALYSIS**

**Data Flow Diagrams**

**Data Dictionary**

**Structured English**

## **DESIGN**

**Structure Charts**

**Documents**

**Module Descriptions**

**Interfaces**

**Performance**

## **IMPLEMENTATION**

**Coding Conventions**

**Coding Practices**

**Error Handling**

**Messages**

**Listing Formats**

**Language Efficiencies**

CONTROL DATA PRIVATE

# ENFORCEMENT

- **Management**  
**Design Team**
- **Automated Tools**  
**SES**
- **Peers**  
**Walkthroughs**  
**Code Reviews**
- **Integration and Evaluation**  
**Quality Assurance**

CONTROL DATA PRIVATE

**II**

**SIS**

CONTROL DATA PRIVATE



# **SIS PRIORITIES**

- 1. Usability/Human Engineering**
- 2. Uniformity/Consistency**
- 3. Good Performance**
- 4. C170 Compatibility**



# INPUT

- **System Command Language (SCL)**
- **Product Call Parameters**
- **Source Input**
- **Product Directives**

CONTROL DATA PRIVATE

# PRODUCT CALL PARAMETERS

## RULES

- If the parameter exists in the SIS, use it.
- SIS parameters cannot be used for undefined purposes.
- You do not have to use all the parameters.
- New parameters must be approved.

## GUIDELINES

- Consider new options instead of new parameters.
- Use names that emphasize relationships.



# DIRECTIVES

## COMPILATION

**eject**  
**list/nolist**  
**space =**  
**title =**  
**subtitle =**  
**range/norange**  
**trace/notrace**  
**debug/nodebug**  
**sequence/nosequence**  
**objlist/noobjlist**  
**prefix =**

## PRODUCT

**brief/full**  
**count**  
**file**  
**wait/nowait**  
**user/password**  
**upon**  
**library**

CONTROL DATA PRIVATE



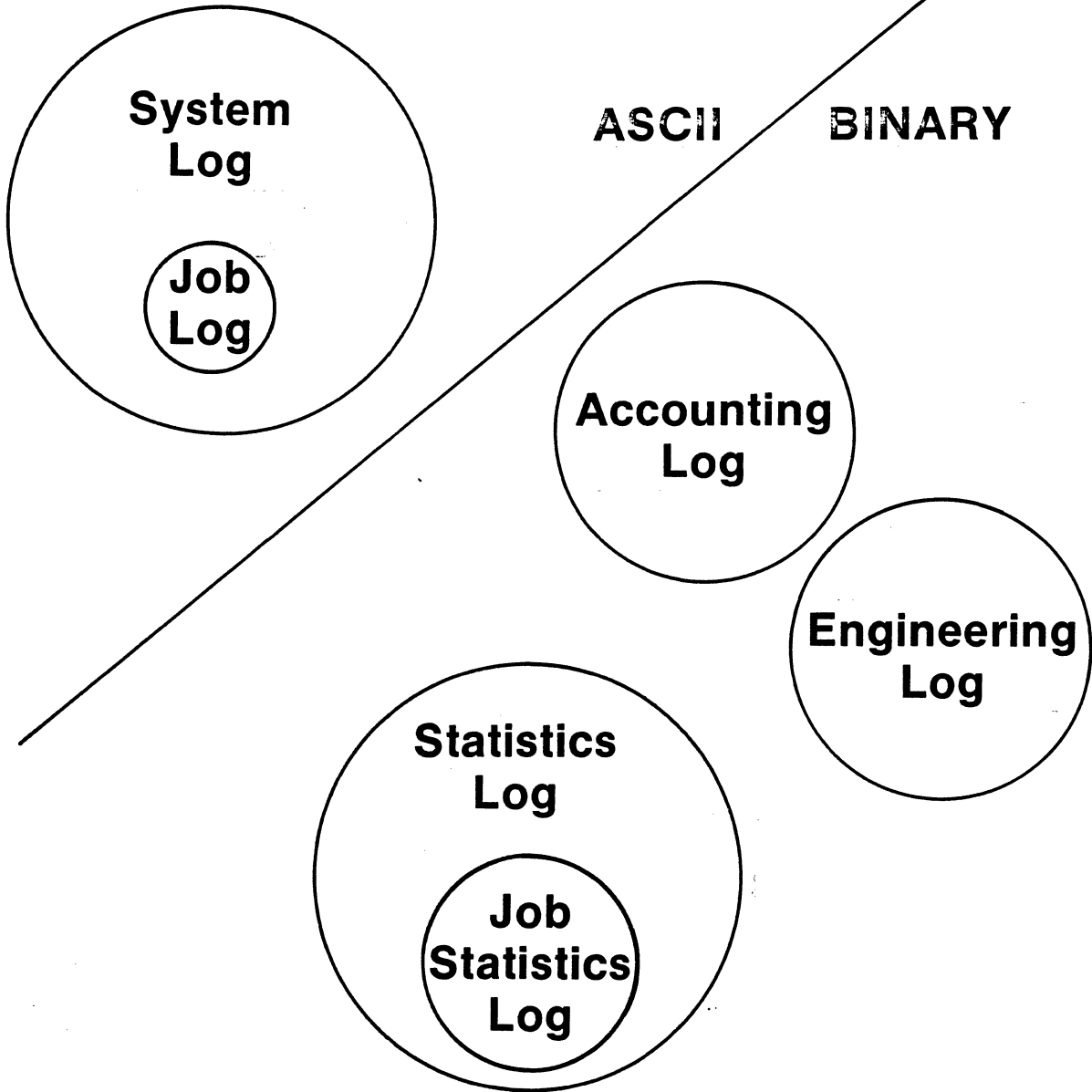
# OUTPUT

- **NUMBER BASES**
- **OUTPUT LOGS**
- **LISTABLE OUTPUT**
- **USAGE STATISTICS**

CONTROL DATA PRIVATE

# OUTPUT LOGS

SIS  
3



CONTROL DATA PRIVATE

# LISTABLE OUTPUT



- **Vertical Layout**
- **Narrow/Wide Format**
- **Source Listing**
  - Object Code**
  - Map**
  - Cross-Reference**
  - Error Listing**

CONTROL DATA PRIVATE

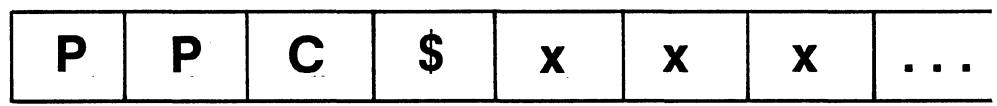


# **SYSTEM-WIDE CONVENTIONS**

- **Names, Dates, and Times**
- **Interactive Processing**
- **Installation Parameters**
- **Error Processing**
- **Effective Use of Hardware**

CONTROL DATA PRIVATE

# NAMES



↓  
**Meaningful Name**

↓  
**CLASS**

- C** — Constant
- F** — File
- P** — Procedure
- V** — Variable
- and so on

↓  
**PRODUCT ID**

- AM** — Access Method
- CY** — CYBIL
- FT** — FORTRAN
- JM** — Job Management
- OS** — Operating System
- PF** — Perm. File Manager
- and so on

CONTROL DATA PRIVATE

# INTERACTIVE COMMUNICATION

## MESSAGES

- Courteous
- Understandable
- Short form

## LISTING

- Levels of detail
- 72 or 132 char/line formats
- No loss of data

## INPUT

- Easily correctable errors
- Reduce typing
- Flexible sources
- Flexible modes

## CONNECT/DISCONNECT

- Terminal  $\equiv$  ASCII sequential file
- Logical disconnect
- Interruptible processes
- Restart after break

## STATUS TO USER

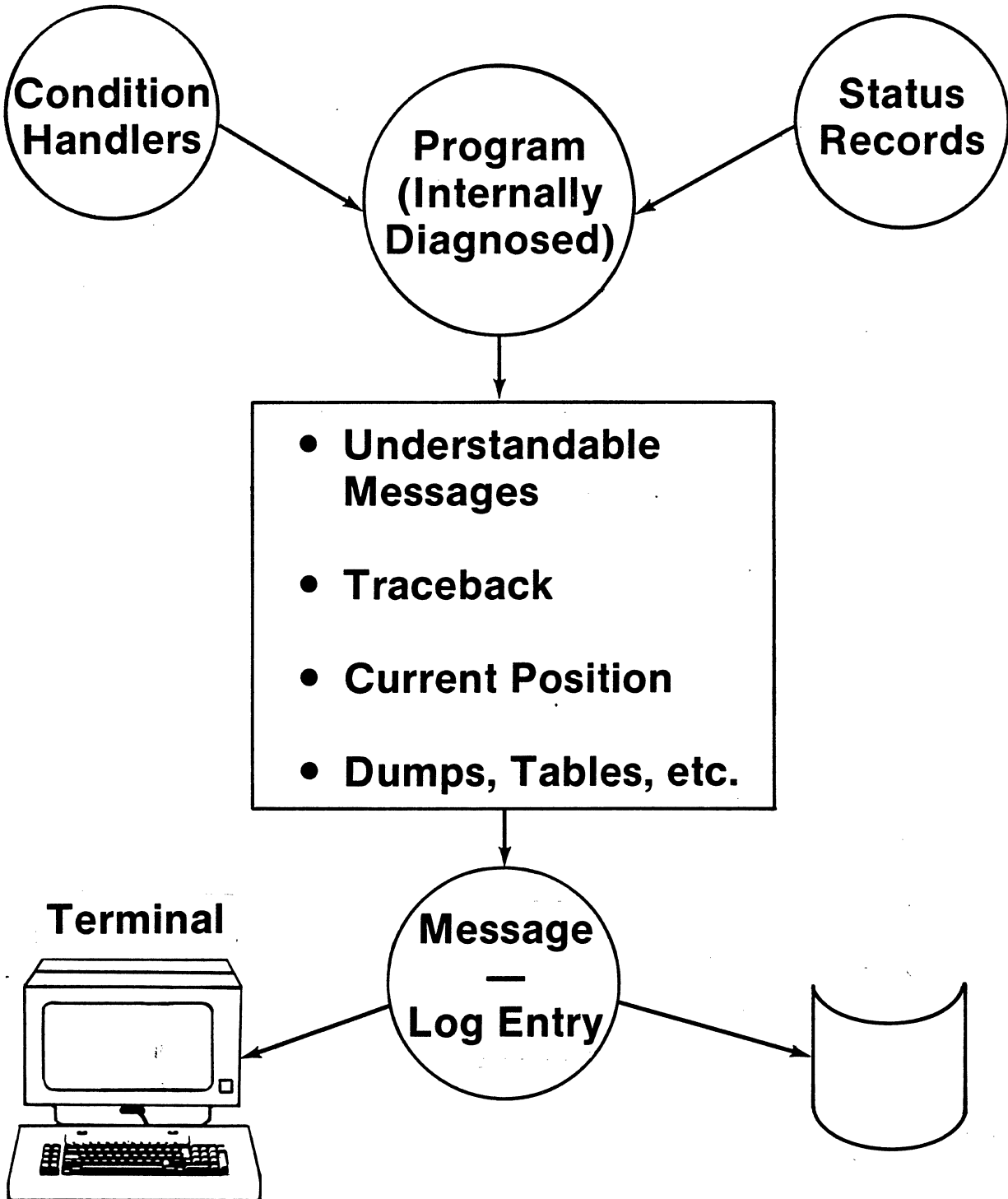
- Almost any time
- Progress reports
- Environment/resources
- Real-time reporting

## HELP

- A reasonable response anytime

CONTROL DATA PRIVATE

# ERROR PROCESSING







# **EFFECTIVE USE OF HARDWARE**

## **HARDWARE OPERATION**

**Register reservation  
Alignment**

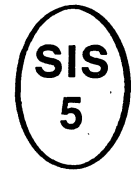
## **PERFORMANCE**

**Locality  
Register use**

## **SECURITY**

**Use callers pointer  
Avoid passing pointers**

CONTROL DATA PRIVATE

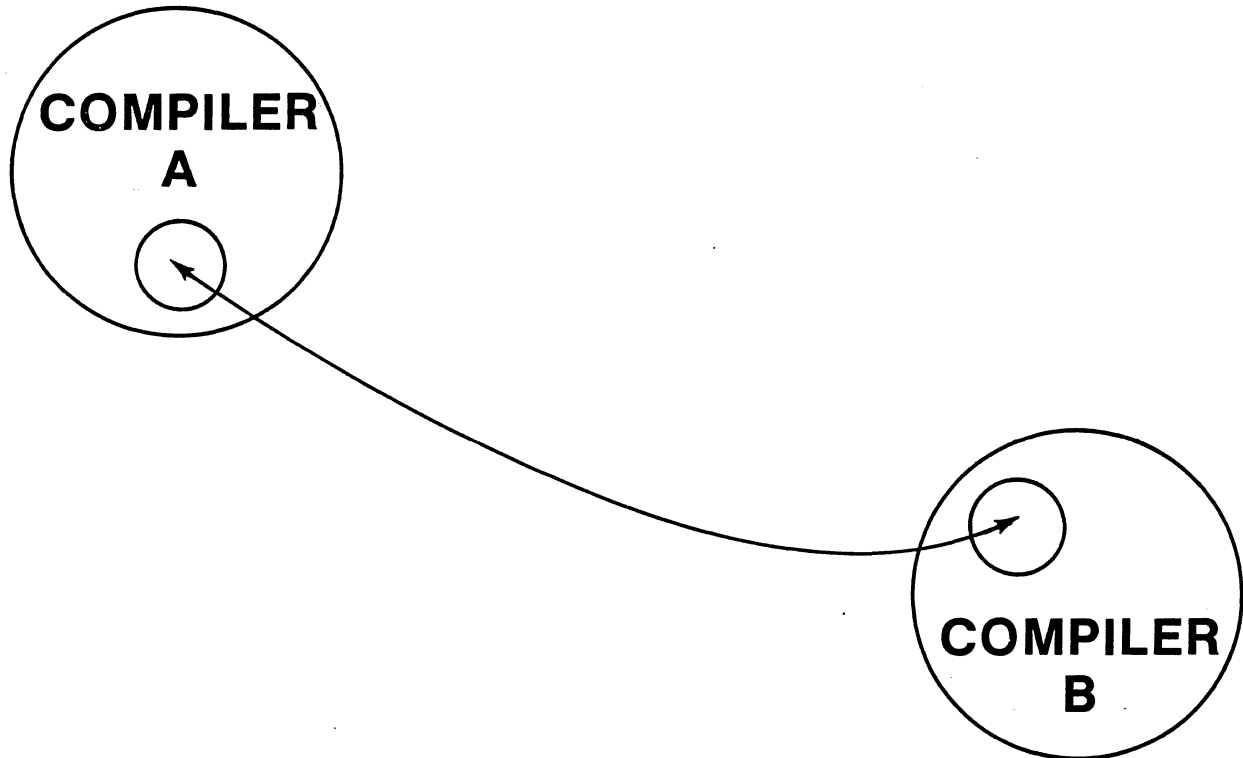


# **COMPILER AND ASSEMBLER OBJECT CODE**

- **Use of Loader Features**
- **Interlanguage Calling Sequences**
- **Storage Management**
- **Common Support Modules**

CONTROL DATA PRIVATE

# INTERLANGUAGE CALLING SEQUENCE



- **Information Required Across Call**
- **Parameter Lists**
- **Data Representation**
- **Data Mapping**

CONTROL DATA PRIVATE

# COMMON COMPILER MODULES

## DIAGNOSTIC MESSAGES

CCP\$sdm\_set\_diagnostic\_mode  
CCP\$ddl\_declare\_diagnostic\_level  
CCP\$rsd\_reset\_diagnostics  
CCP\$iin\_insert\_name  
CCP\$iad\_issue\_a\_diagnostic  
CCP\$res\_return\_error\_severity

## LISTABLE OUTPUT

CCP\$fsl\_format\_source\_line  
CCP\$foh\_format\_heading  
CCP\$fol\_format\_output\_listing

## SYMBOL TABLE FOR DEBUG PACKAGE

## STORAGE MAP/ATTRIBUTE/ CROSS-REFERENCE LISTS

CCP\$den\_define\_entity  
CCP\$der\_define\_entity\_reference  
CCP\$fam\_format\_attribute\_map  
CCP\$iat\_insert\_attribute\_token

## USAGE STATISTICS

## REPRIEVE STATISTICS

CONTROL DATA PRIVATE



# **COMMON PROCEDURES**

## **NOS/VE PROGRAM INTERFACE**

**SCL Parameter Processing**  
**Message Generator**  
**Condition Handling**  
**Time, Date, Job Name, and so on**  
**Logs**  
**Status Interrogation**

## **MATH LIBRARY (CMML)**

**Math Routines**  
**Numeric Conversion**

## **COMMON CODE GENERATOR (CCG)**

## **MEMORY MANAGEMENT (CYBIL)**

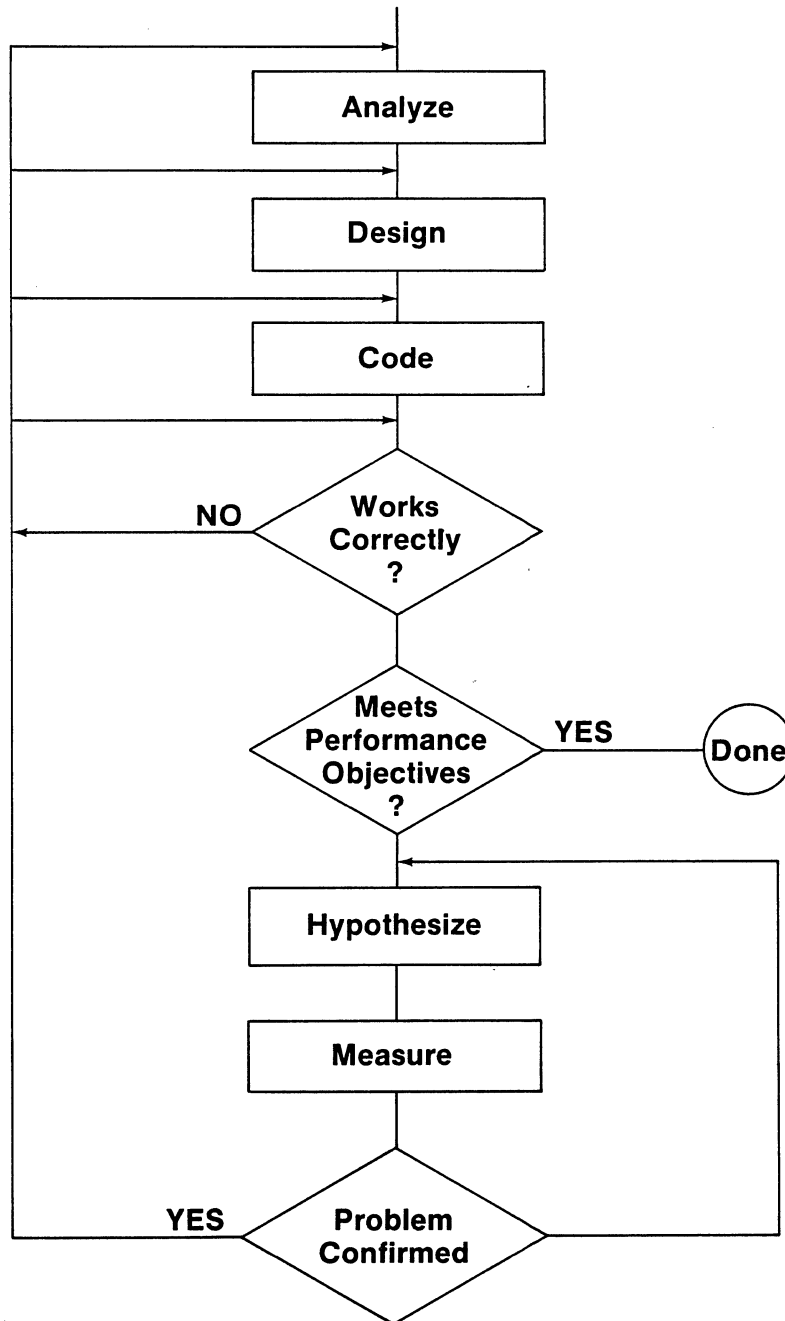
CONTROL DATA PRIVATE



# PERFORMANCE

CONTROL DATA PRIVATE

# IMPLEMENTATION PROCESS



CONTROL DATA PRIVATE

# **GOOD DESIGN BEGETS GOOD PERFORMANCE**

- **Correct**
- **Fast**



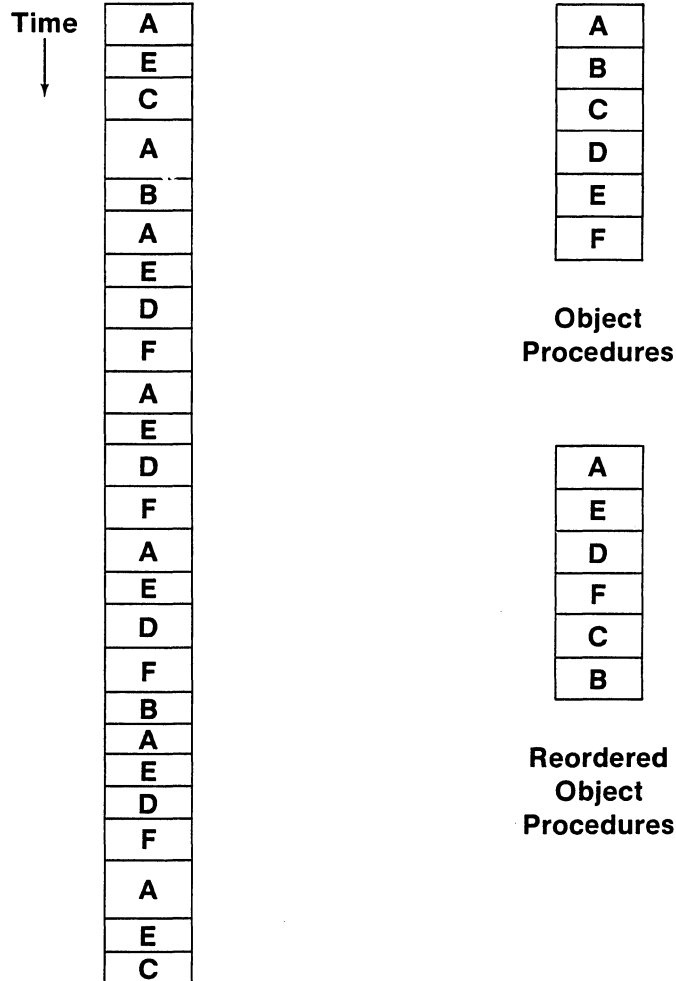
# **CODE FOR SIMPLICITY AND CLARITY**

- **Simple search techniques**
- **Straightforward interfaces**
- **Document for future reader**
- **Small procedures (10-100 statements)**
- **Avoid overly tight code**
- **Single-purpose procedures**
- **Don't pass control information**
- **Meaningful names**

# MEASURE, THEN REVISE

- **APD to Isolate Problem**
- **Code Considerations**
  - Locality**
  - Algorithms**
  - Language Inefficiencies**
- **Data Considerations**
  - Locality**
  - Referencing Algorithms**
  - Data Structures**

# ANALYZE PROGRAM DYNAMICS



- Small procedures
- Low coupling
- High cohesion
- Exception processing and initialization in separate procedures

CONTROL DATA PRIVATE

# LOCALITY OF REFERENCE

## OBJECTIVES: Minimize

1. Working set variations
2. Page faults
3. Average working set size

## RECOMMENDED PRACTICES

**Low coupling**

**Cluster related data**

**Declare data at lowest level**

**Initialize data as you use it**

**Access data sequentially**

**Access memory, use the data, release memory**

# LOCALITY OF REFERENCE

## **OBJECTIVES: Minimize**

- 1. Working set variations**
- 2. Page faults**
- 3. Average working set size**

## **DISCOURAGED PRACTICES**

**Large numbers of segments**

**Use of static chain**

**“Elaborate” search/sort techniques**

**Interleaved structures**

**Externally declared variables**

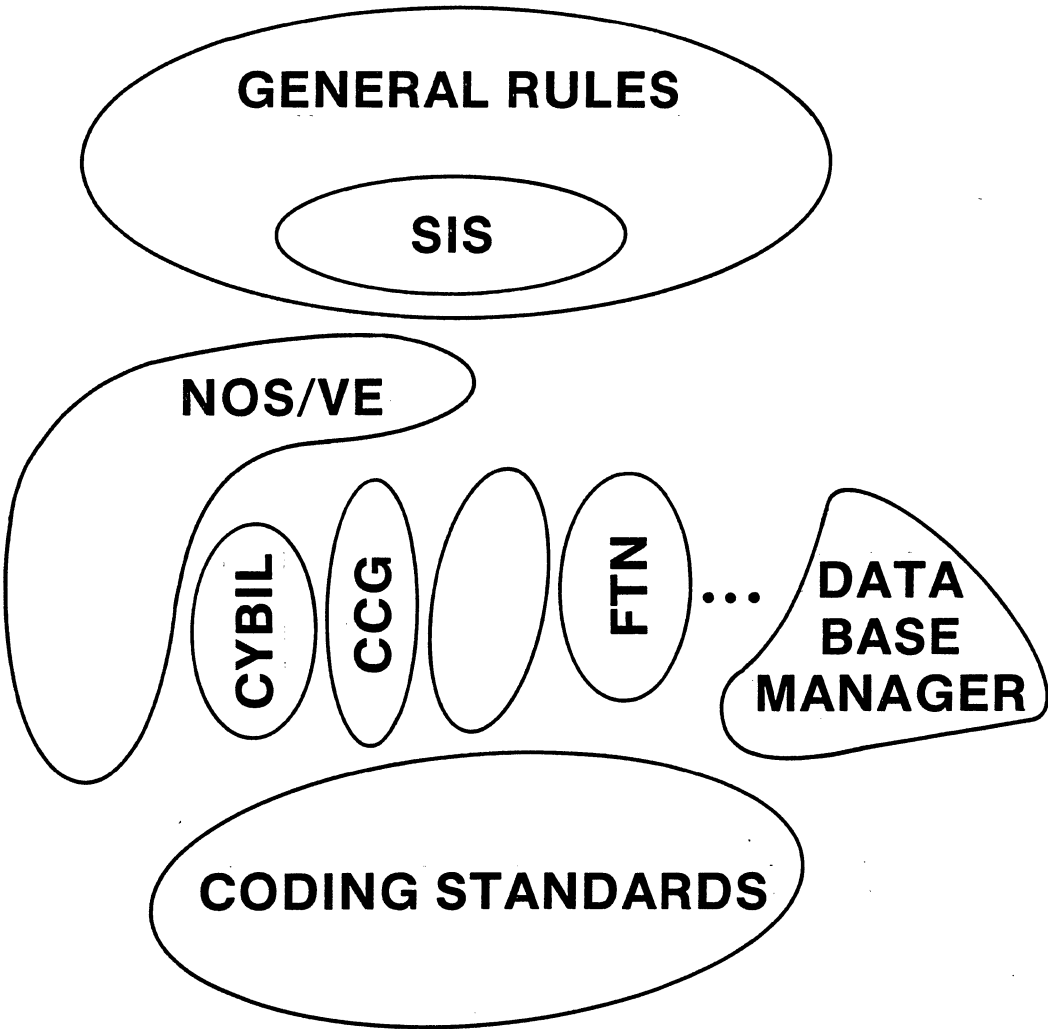
**Trying to trick the paging mechanism**

# IV

## PROJECT CONVENTIONS

CONTROL DATA PRIVATE

# GUIDELINES HIERARCHY



CONTROL DATA PRIVATE

# PROJECT GUIDELINES

- **Mode of Operating**
  - Review cycles**
  - Change procedures**
  - Analysis/design tools**
  - Documents**
  - Development/test tools**
- **Programming Language**
  - Naming**
  - Layout**
  - Efficiencies**
  - Interfaces**
  - Techniques**
- **Packaging**
  - Size**
  - Common decks**
  - Performance**

CONTROL DATA PRIVATE



# **NOS/VE**

**Design Team**  
**Document Organization**  
**Procedural Interface**  
**Program Library Conventions**  
**CYBIL Coding Convention**  
**Code Submittal Process**  
**Document Maintenance**  
**Yourdon Methodology**

---

CONTROL DATA PRIVATE

# PARAMETER TYPING

- **Use type identifiers.**
- **Use self-documenting feature of ordinals.**
- **Use constants to:**
  - **delimit subranges**
  - **specify string length**
- **Use sets to specify multiple subfunctions.**
- **Arrays are good if:**
  - **multiple generation or manipulation will take place**
  - **components are independent**
- **Record should be a unified entity.**
  - **field has a clear relationship**
  - **all fields are essential to the function being performed**
  - **one directional**
- **Avoid packed structures, adaptable type, and bound variant records.**

# **NOS/VE CYBIL CODING CONVENTION**

- **Formatter**
- **Use of CYBIL**
- **Use of English**
- **CYBIL Naming Convention**
- **Module and Procedure Documentation**
- **Title**
- **Commenting**
- **Attribute Comments**
- **Module Organization**

# USE OF CYBIL

## READABILITY AND CLARITY

- Label both ends of structured statement.
- Use ordinal and subrange.
- Declare all input parameters first.
- Parameters to procs ① pass information ② document data used by the proc.
- Use procedures ① as subroutines ② to show structure.

## RELIABILITY AND SAFETY

- Cover all CASEs.
- Don't use default parameter values.
- Use parentheses in arithmetic statements.
- Avoid #LOC.

## PERFORMANCE

- Avoid XREF/XDCL.
- Declarations should be at lowest level.
- The first condition on a boolean expression should be the most probable.
- Consider PUSH instead of ALLOCATE/FREE.

# CCG CODING CONVENTION

- **Philosophy**
  1. **Code correct**
  2. **Sensible structure**
  3. **Pretty**
- **Module Layout**
  1. **Use of pragmat**
  2. **Commenting**
  3. **Procedure parameters**
- **Common Decks**
- **Declarations**
- **CYBIL Formatter**

# **PACKAGING GUIDELINES**

## **MODULE SIZE AND CONTENTS**

- **Limit scope of declarations**
- **Localize static data**
- **Repackaging ease**

## **COMMON DECKS**

- **Don't hand-code the same thing in several places — use a procedure**
- **Self-contained**

## **PROCEDURE SIZE AND CONTENTS**

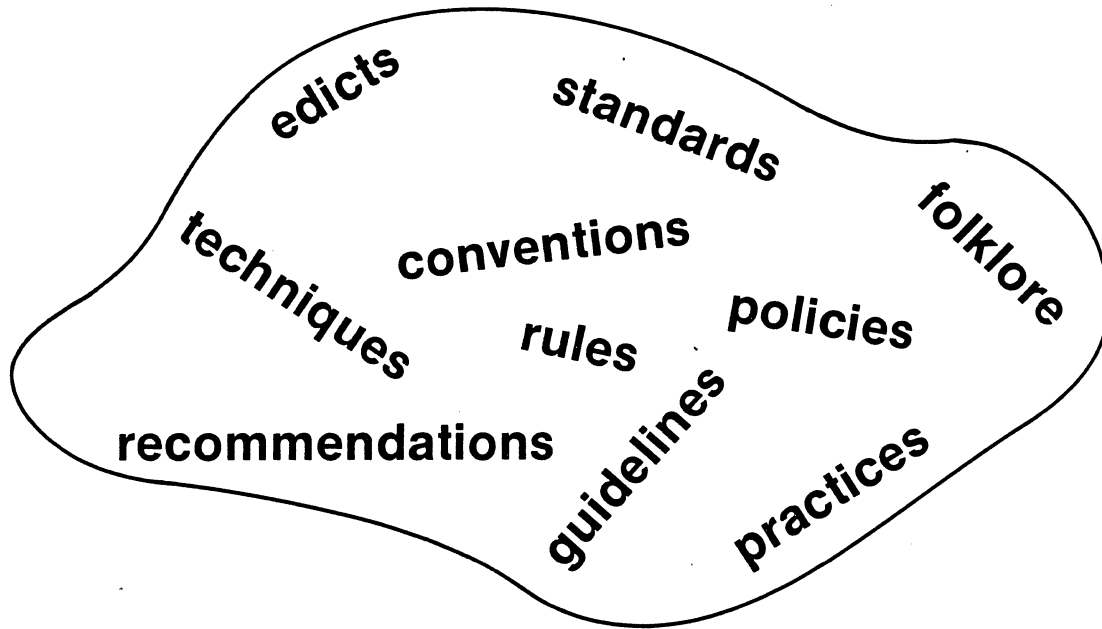
- **Single purpose, testable separately**
- **Block comments instead of one-liners**  
**10—100 statements?!**

## **MANAGEMENT**

- **Source code**
- **Object code**
- **Library**

CONTROL DATA PRIVATE

# SUMMARY



1. Where are they?
2. What kinds of things do they cover?
3. Which ones apply to me?

# APPENDIX

CONTROL DATA PRIVATE



# GENERAL STANDARDS

AGREE?	RULE	STANDARD	GUIDELINE

**What guidelines, conventions, and standards would you want and expect to have specified for any software development project you become involved with?**

# SIS CONVENTIONS

Compare this set of declarations with the set on the next page. List the conventions used. What do you like or dislike about each?

## NOS/VE:

```
{ NOS/180 request status record: used to convey results of all system }  
{ requests and optionally all commands. }
```

### TYPE

```
ost$status = record  
  normal: boolean,  
  state: ost$status_states,  
  identifier: string (2),  
  subidentifier: string (3),  
  condition: ost$status_condition,  
  subcondition: ost$status_subcondition,  
  text: clt$char_string,  
  recend;  
  
ost$status_states = (osc$normal_status, osc$informative_status,  
  osc$warning_status, osc$error_status, osc$fatal_status),  
  
ost$status_condition = 0 .. osc$max_condition,  
  
ost$status_subcondition = 0 .. osc$max_subcondition,  
  
clt$char_string = record  
  lhi: 1 .. 257,  
  rhi: 0 .. 256,  
  buf: string (256),  
  recend;
```

### CONST

```
osc$max_condition = 16000,  
osc$max_subcondition = 100;
```

```
{ Asynchronous request parameter: used by all NOS/180 requests that }  
{ can be performed asynchronously to indicate whether the caller }  
{ wishes to execute the request synchronously or asynchronously. }
```

### TYPE

```
ost$wait = (osc$wait, osc$nowait);
```

```
{ Secure memory/file parameter }
```

### TYPE

```
ost$clear_file_space = boolean;
```

CONTROL DATA PRIVATE

## COMMON CODE GENERATOR:

{ SPECIFIED CONSTANTS AND TYPES:

{ The remainder of the constants and types are both described  
{ and defined (in terms of value) in the CCG180 Interface  
{ Specification.

CONST

  cgc\$\_max\_section\_offset = 7fffffff(16)  
  { the maximum offset within a section of any item } ;

TYPE

  cgt\$\_byte\_offset = 0 .. cgc\$\_max\_section\_offset,  
  { a byte offset

  cgt\$\_byte\_length = 0 .. cgc\$\_max\_section\_offset + 1,  
  { maximum section length

  cgt\$\_number = - 800000(16) .. 7fffff(16),  
  { a CCG180 "identifier"

  cgt\$\_interf\_class = 0 .. 15,  
  { the interference class values

  cgt\$\_library\_name = string (31),  
  { standard name field

  cgt\$\_optimization\_level = (  
  { optimization level ordinal  
  cgo\$\_opt\_level\_local, cgo\$\_opt\_level\_global),

  cgt\$\_host\_compiler = (  
  { ordinal over the possible hosts  
  cgo\$\_host\_algol, cgo\$\_host\_algol\_68, cgo\$\_host\_basic, cgo\$\_host\_cobol,  
  cgo\$\_host\_fortran, cgo\$\_host\_oblige, cgo\$\_host\_pascalx, cgo\$\_host\_pascal,  
  cgo\$\_host\_pl\_i, cgo\$\_host\_symbol),

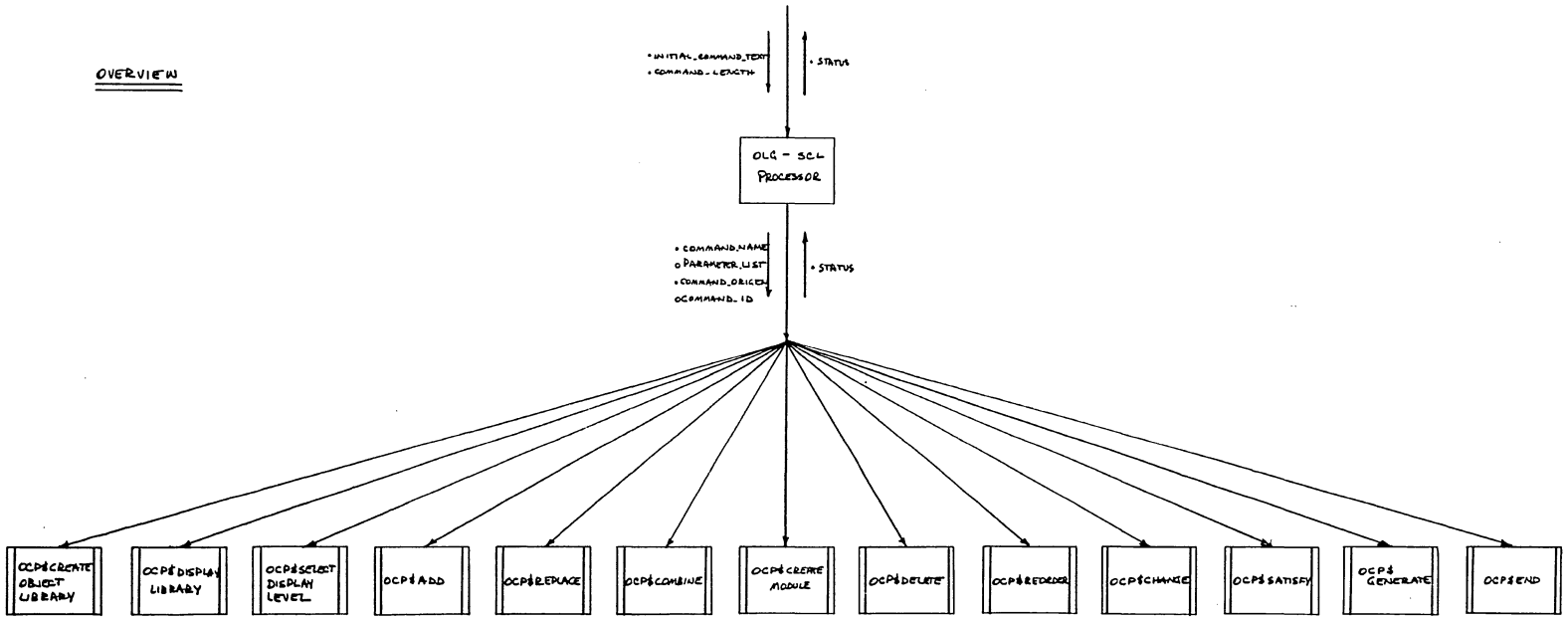
  cgt\$\_section\_access = (  
  { the access attributes for data areas  
  cgo\$\_access\_read, cgo\$\_access\_write),

  cgt\$\_sections = (  
  { varieties of loader sections  
  cgo\$\_sect\_common, cgo\$\_sect\_working, cgo\$\_sect\_ext\_common,  
  cgo\$\_sect\_ext\_working),

  cgt\$\_data\_areas = (  
  { varieties of data areas

CONTROL DATA PRIVATE

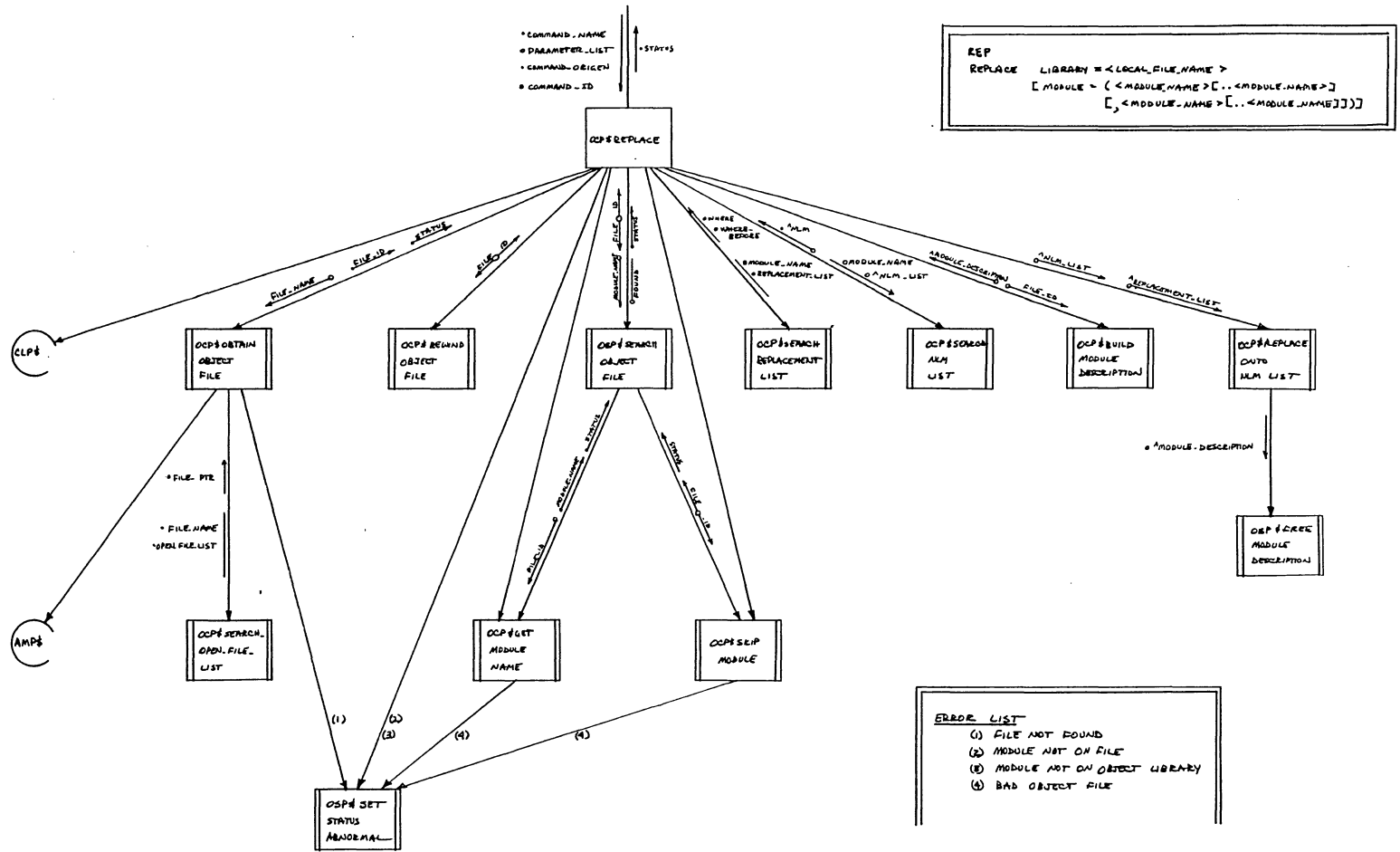
OVERVIEW



CONTROL DATA PRIVATE

CONTROL DATA PRIVATE

48



# FTN 5 CROSS-REFERENCE LISTING FORMATTED BY CCM

IDENTIFIER	DEFINED UN	TYPE	ATTRIBUTES REFERENCES	M=MODIFY, A=ATTRIBUTE, S=SUBSCRIPT, I=I/O REF, R=READ, W=WRITE, P=PARAM
I	39	SIMPLE#VAR	SIZE 1 WORD, OFFSET 7701 BYTES ,INTEGER 40 41 45 46 46 48 49 50 50 51	
ID	2	ARRAY	SIZE 210 WORDS, OFFSET 1232 BYTES ,INTEGER 10/M 54/M 55/M 63	
IJ	84	SIMPLE#VAR	SIZE 1 WORD, OFFSET 7715 BYTES ,INTEGER 84 84	
IM	27	SIMPLE#VAR	SIZE 1 WORD, OFFSET 7677 BYTES ,INTEGER 60	
IMQN	2	ARRAY	SIZE 210 WORDS, OFFSET 7350 BYTES ,INTEGER 17/M 56/M 57/M 59/M 62 72	
IMT	2	ARRAY	SIZE 210 WORDS, OFFSET 2742 BYTES ,INTEGER 11/M 35/M 36/M 40/M 42 42	44 46/M 49 61 62
IN	28	SIMPLE#VAR	SIZE 1 WORD, OFFSET 7700 BYTES ,INTEGER 70	
IQ	2	ARRAY	SIZE 210 WORDS, OFFSET 7026 BYTES ,INTEGER 16/M 33/M 34/M 79	
IR	2	ARRAY	SIZE 840 WORDS, OFFSET 4774 BYTES ,INTEGER 42 43 44 47 61 62 73 74 78 79 80	63 66 67 71 72
IS	2	ARRAY	SIZE 210 WORDS, OFFSET 6504 BYTES ,INTEGER 15/M 31/M 32/M 78	
ISAM	2	ARRAY	SIZE 210 WORDS, OFFSET 1554 BYTES ,INTEGER 18/M 67 69 74 76 79	80
ISON	2	ARRAY	SIZE 210 WORDS, OFFSET 3606 BYTES ,INTEGER 8/M 44 46 47 50 61 69/M 71 72 73 74 78	62 63 65 66 67
ISON1	2	ARRAY	SIZE 210 WORDS, OFFSET 4130 BYTES ,INTEGER 13/M 49/M 67 78 80 81	62 83 84/M
ISON2	2	ARRAY	SIZE 210 WORDS, OFFSET 4452 BYTES ,INTEGER 14/M 51/M 74 76/M 79	
IM	23	SIMPLE#VAR	SIZE 1 WORD, OFFSET 7674 BYTES ,INTEGER 25 27 28 29 29/M 42 63 66 67 71 72 73 84	43 44 47 61 62 74 78 79 80 84
IZ	5	SIMPLE#VAR	SIZE 1 WORD, OFFSET 7672 BYTES ,INTEGER 6 7 8 9 10 11 17 18	12 13 14 15 16
J	64	SIMPLE#VAR	SIZE 1 WORD, OFFSET 7707 BYTES ,INTEGER 65 65 68 69 69 75	76 76
JNT	2	ARRAY	SIZE 210 WORDS, OFFSET 3264 BYTES ,INTEGER 12/M 37/M 38/M 41/M 43 43 73	47 50/P 51 71 72
K	2	ARRAY	SIZE 210 WORDS, OFFSET 710 BYTES ,INTEGER 9/M 52/M 53/M 63 62/M 66	73
M	2	ARRAY	SIZE 210 WORDS, OFFSET 2076 BYTES ,INTEGER 6/M 42 44 61	
N	2	ARRAY	SIZE 210 WORDS, OFFSET 2420 BYTES ,INTEGER 7/M 43 47 71	
NA	81	SIMPLE#VAR	SIZE 1 WORD, OFFSET 7713 BYTES ,INTEGER 82 83	
NB	82	SIMPLE#VAR	SIZE 1 WORD, OFFSET 7714 BYTES ,INTEGER 83 84/M	
NC	23	SIMPLE#VAR	SIZE 1 WORD, OFFSET 7675 BYTES ,INTEGER 26 29 30 30/M 42 43 66 67 71 72 73 74	44 47 61 62 63 78 79 80 83/M 84/M
NCI	26	SIMPLE#VAR	SIZE 1 WORD, OFFSET 7676 BYTES ,INTEGER 32 34 36 38 53 55	57

CONTROL DATA PRIVATE

# SAMPLE FTN 5 CROSS-REFERENCE LISTING

--VARIABLE MAP-- (LO=A/K)							A=ARGLIST, C=CTRL OF DO, I=DATA INIT, R=READ, S=STORE, U=I/C UNIT, W=WRITE												
NAME	ADDRESS	BLOCK	PROPERTIES	TYPE	SIZE	REFERENCES													
I	7701B			INTEGER		39/C 40 41 42/C 46 46 48/C 49 49 50													
IU	1232B			INTEGER	210	50 51 51 58/C 59 60/C 70													
IJ	7715B			INTEGER		2 10/S 54/S 55/S 63													
IM	7677B			INTEGER		84 84/C 84													
IMUN	735Jd			INTEGER	210	27/S 60/C													
INT	2742B			INTEGER	210	2 17/S 56/S 57/S 55/S 62 72													
						2 11/S 35/S 36/S 40/S 42 42 44 46/S 49													
						61 62 66													
IN	7700B			INTEGER		28/S 70													
IU	702oB			INTEGER	210	2 16/S 33/S 34/S 79													
IR	4774B			INTEGER	840	2 42 43 44 47 61 62 63 66 67													
						71 72 73 74 78 79 80													
IS	6504B			INTEGER	210	2 15/S 31/S 32/S 78													
ISAM	1554B			INTEGER	210	2 18/S 67 69 74 76 79 80													
ISUM	360oB			INTEGER	210	2 8/S 64 46 47 50 61 62 63 65													
PROGRAM PP 74/74 CPT=0							FTN 5.0+508			11/07/79 15.59.51			PAGE 3						
NAME	ADDRESS	BLOCK	PROPERTIES	TYPE	SIZE	REFERENCES													
ISUM1	4130B			INTEGER	210	66 67 69/S 71 72 73 74 78 80													
ISUM2	4452B			INTEGER	210	2 13/S 49/S 67 78 80 81 82 83 84/W													
IM	7674B			INTEGER		2 14/S 51/S 74 76/S 79													
						23/R 25 27 28 29 29/S 42 43 44 47													
						61 62 63 66 67 71 72 73 74 78													
						79 80 84 84/C													
IZ	7672B			INTEGER		5/C 6 7 8 9 10 11 12 13 14													
						15 16 17 18													
J	7707B			INTEGER		64/C 65 65 68/C 69 69 75/C 76 76													
JNT	3264B			INTEGER	210	2 12/S 37/S 38/S 41/S 43 43 47 50/S 51													
						71 72 73													
K	710B			INTEGER	210	2 9/S 52/S 53/S 63 65/S 66 73													
M	2076B			INTEGER	210	2 6/S 42 44 61													
N	2420B			INTEGER	210	2 7/S 43 47 71													
NA	7713B			INTEGER		81/S 82 83													
NB	7714B			INTEGER		82/S 83 84/W													
NC	7675B			INTEGER		23/R 26 29 30 30/S 42 43 44 47 61													
						62 63 66 67 71 72 73 74 78 79													
						80 83/S 84/W													
NCI	7676B			INTEGER		26/S 32 34 36 38 53 55 57													

CONTROL DATA PRIVATE

# PROJECT STANDARDS

AGREE?	RULE	STANDARD	GUIDELINE

**Suppose you are assigned as manager of a software project (for example, a data management system). What guidelines would you specify for the project? Assume that the SIS must be adhered to.**



# PROJECT CONVENTION

Compare the following procedure declaration with the one on the next page.

List the conventions used.

What do you like or dislike about each?

Cyber 180 Common Code Generator Interface Specification

-----  
12.0 PHYSICAL INTERFACES.  
12.4.2 DEFINITION PROCEDURES  
-----

CGP\$dbt\_define\_bit\_field

a. XREF Declaration

```
PROCEDURE [XREF] CGP$dbt_define_bit_field (  
    base: CGT$_number,  
    field_attributes: CGT$_field_attributes_set,  
    interf_class: CGT$_interf_class,  
    lexical_level: 0..7,  
    byte_offset: CGT$_byte_offset,  
    first_bit_offset,  
    last_bit_offset: 0..63,  
    name: STRING (*)  
    VAR F_number: CGT$_number)
```

b. Function

This procedure defines a bit aligned field, upto 63 bits long. The base of the field is given by the base parameter, its byte offset by the byte\_offset parameter, its bit offset and length by the first\_bit\_offset and last\_bit\_offset parameters and its attributes by the field attributes and interf\_class parameters. Note that bit fields cannot have a bdp type. If the name of the field is required for object code listing, it should be supplied as the name parameter. The procedure returns an F-number that describes it.

CONTROL DATA PRIVATE

# PROJECT CONVENTION

List the conventions used.  
What do you like or dislike about each?

NOS/VE ERS - PROGRAM INTERFACE

3-5

12/17/79

-----  
3.0 RESOURCE MANAGEMENT -  
3.1.1.5 RMP\$DEFINE\_ALLOCATION\_UNIT  
-----

## 3.1.1.5 RMP\$DEFINE\_ALLOCATION\_UNIT

```
(
( The purpose of this request is to define the allocation
( unit size of a file prior to file access.
( This request is ignored if a previous REQUEST command has defined)
( the allocation_unit for the file or if the file already exists.)
( If the request is ignored an abnormal status is returned.)
(
( RMP$DEFINE_ALLOCATION_UNIT (LOCAL_FILE_NAME,
( ALLOCATION_UNIT, STATUS)
(
( LOCAL_FILE_NAME: (input) this parameter specifies the local file
( name of the file for which the request is being issued.
(
( ALLOCATION_UNIT: (input) This parameter specifies the number of
( contiguous mass storage device allocation units which are)
( allocated to the file each time the system determines that)
( allocation is necessary.)
( Allocation_unit options are:
(
( rmc$default_au - specifies system default (a1)
( rmc$a1 - 1 device allocation unit (DAU)
( rmc$a2 - 2 DAUs
( rmc$a4 - 4 DAUs
( rmc$a8 - 8 DAUs
( rmc$a16- 16 DAUs
( rmc$a32- 32 DAUs
(
( STATUS: (output) This parameter specifies the request status.
(
```

```
PROCEDURE [XREF] rmp$define_allocation_unit (local_file_name:
amt$local_file_name;
allocation_unit: rmt$allocation_unit;
VAR status: ost$status);
```

```
*callc andname
*callc rmdau
*callc osdstat
```

CONTROL DATA PRIVATE

# ALGORITHMS AND CODING PRACTICES\*

The following recommendations are for the FTN/180 implementation process. They are in no particular order beyond a loose attempt to separate them into “general coding” and “data reference” categories. Their common aim is to improve locality of reference and **main** memory usage during some short time span (on the order of 1 to 100 milliseconds).

1. Reduce the short-term use of main memory, even if it causes long-term virtual memory usage to increase. Main memory is expensive; virtual memory (auxiliary page storage) is practically free.
2. Write in-line code for the normal, average, standard cases. Move special, pathological, end-case code out-of-line, possibly into separate procedures. Remember that the in-line code must **detect** the funny cases, even though it relies on/calls the out-of-line code for the actual processing. Structured programming principles should not be disregarded completely, but an occasional bend of the rules might not hurt much.
3. Don't write overly tight code just to save a few bytes. It tends to be unreliable, nasty to unravel, and even harder to fix. If the bytes really are important, or it's an inner-inner loop, maybe a simpler algorithm would do the same function—clearly.
4. If a heavily-used procedure routinely calls a distant utility routine, consider the possibility of replicating the utility code either inside or near (same page) the procedure.
5. Resist all temptation to stuff too many functions in a procedure, or to fudge on its interface with another procedure, just to save a little memory. Keep in mind that memory is cheap, but PSRs aren't.
6. Try to confine references (either data or code) to pages that should be in main memory simultaneously, that is, in the current working set. Remember that the virtual addresses may be widely separated, yet may refer to pages that are adjacent in main memory at the moment.

\*Reprinted with permission from “**Impact of C180 Virtual Memory of FTN 180,**” Dillion, D.C.

CONTROL DATA PRIVATE

7. Don't try to outwit the NOS/180 page management strategy. While it may be entirely possible to do so, such practices as dummy procedure calls "just to drag the next page in early" will probably interfere with more global attempts to fine-tune the compiler's or NOS/180's performance. Also, compiler performance statistics would be artificially distorted.
8. Don't initialize a large number of data areas en masse (e.g., at the beginning of a pass). This could cause many pages to be brought into main memory long before they are really needed. Instead, initialize each data area just before it is first used. This dictum can be ignored if the data areas are small and located in the same page.
9. Don't reuse global shared ("common") storage for different purposes during separate phases of compilation. On a virtual machine, the technique probably won't save any main memory. Furthermore, the global coupling is often not obvious and can cause very subtle bugs.
10. Where possible, process data and release its storage in small chunks, not large ones. This may require delicate compromises between code space and data space.
11. Don't over-pack data structures just because PASCAL makes it easy. The compiler code to pack and unpack the structures is bulky and slow, and could degrade performance more than a slight increase in data paging. Review each major data structure and make a reasoned decision about packing the structure. When hardware becomes available for performance testing, conduct tests to update the decisions.
12. "Reference data in the order in which it is stored and/or store data in the order in which it is referenced. This is particularly true of arrays. If an array is stored by columns (as in FORTRAN), complete all references to a single column before moving to the next. The order in which data areas are referenced is, of course, of no consequence if the entire area fits into a single page. Most page-replacement algorithms tend to favor pages that have been used recently. Therefore if a procedure causes a large sequential space of storage to be traversed, the direction of scan should be reversed in alternate passes." [Morrison] (The reversing technique may strain the spirit of paragraph 7., "thou shalt not trick the paging strategy." It depends on the exact circumstances, and one's own conscience.)
13. "Avoid the use of elaborate search strategies for large data areas. Avoid the use of large, linked lists if these techniques cause a wide range of addresses to be referenced. Methods of using list structures are referenced. The use of binary search for sequential tables spanning many pages should be carefully evaluated. Useful alternatives to binary search are hashing entries for direct access, or resequencing the table by frequency of use so that a sequential search may be used." [Morrison]

CONTROL DATA PRIVATE

14. Reference a data structure naturally—which means, if possible, sequentially. Consider the possibility of reordering the structure if the reference pattern suggests excessive paging demands. For example, consider the multiplication of two matrices. This requires referencing one matrix in row order and the other in column order. Depending on the order of element storage, one of these reference patterns will cause excessive paging if the matrix spans many pages (the problem does not exist if each matrix fits in one page). The problem may be minimized by transposing the offending matrix before the multiply. The transpose, or reordering, operation may require less time than the paging overhead for the ill-conditioned case.

Sometimes, one procedure of a program will refer to a data structure in a pattern that differs significantly from others found elsewhere in the program. Either reordering the data structure or revising the referencing algorithm of the lone procedure may improve the pattern.

# REFERENCES

- Architectural Objectives and Requirements, ARH 1688. Control Data Document.
- Boehm, B.W., Brown, J.R., and Lipow, M. "Quantitative Evaluation of Software Quality," from the Second International Conference on Software Engineering, October, 1976. p. 592.
- Brooks. "The Mythical Man-Month," Addison-Wesley, 1975.
- C180 Common Compiler Modules (CCM), Interface Specification. S2987. Control Data Document.
- C180 Mathematical Library (CMML), S2929. Control Data Document.
- C180 System Interface Standard, S2196. Control Data Document.
- "CCG 180 - Coding Conventions," July 11, 1979. Control Data Document.
- CMML Assembly - Language Support System, S3410. Control Data Document.
- CYBIL Implementation Dependent Handbook, ARH3078. Control Data Document.
- Dillon, D.C. "Impact of C180 Virtual Memory on FTN/180."
- Dillon, D.C. and Waddell, J.P. "CYBER 180 Compiler Architecture Guidelines," December 20, 1978. Control Data Document.
- Kernighan and Plauger. "Software Tools," Addison-Wesley, 1976.
- Kernighan and Plauger. "The Elements of Programming Style."
- Morrison, J.E. "User Program Performance in Virtual Storage Systems," IBM Systems Journal, Vol. 12, #3, 1973.
- Myers. "The Art of Software Testing," New York, 1979.
- NOS/VE Project Procedures and Conventions. Control Data Document.
- Rogers, J.A. "Structured Programming for Virtual Storage Systems," IBM Systems Journal, Vol. 14, #4, 1975.

CONTROL DATA PRIVATE

# REFERENCES — Continued

“SYMPL Coding Standard for the SYMPL Project.” Control Data Document.

Wilson, J.A. “Segment Usage,” October 30, 1979.

Weinberg. “The Psychology of Computer Programming,” New York.

CONTROL DATA PRIVATE