

Structured CICS COBOL II Programming Standards

April 2004

Structured CICS COBOL II Programming Standards
Copyright © 2004 by:

Iowa State University
Administrative Technology Services
2nd Floor ASB
Ames, IA 50011-3601

Past and present employees of the Iowa State University (ISU) Administrative Technology Services (ATS) have helped develop and maintain these standards.

All rights reserved. No part of this manual may be reproduced in any form or by any means without the permission in writing from ISU Administrative Technology Services.

Disclaimer

Although these programming standards have been tested and used by the contributors and ISU Administrative Technology Services, neither the contributors, ISU Administrative Technology Services, ISU, nor the State of Iowa, express or imply any warranty as to the accuracy and functioning of these standards, any program developed using these standards, or any related material. Distribution does not constitute any such warranty; neither the contributors, ISU Administrative Technology Services, ISU, nor the State of Iowa assume responsibility in connection with this distribution.

It is understood that no claim will be made against the contributors, ISU Administrative Technology Services, ISU, or the State of Iowa in connection with these standards or with related material.

Contents

Acknowledgement	i
Introduction	ii
Rationale for Standards.....	iii
Purpose of This Handbook	iv
Testing Procedures and Requirements	v
Pre-Production Procedures	v
Production Implementation Procedures	v
Sign-On and Routing Approach to CICS Application Development	1
Sign-On	1
The MENU Program	1
Standard Screen Formats	1
General Guidelines for Screen Definition	2
System Menu Screens for Function Selection.....	2
Display/Add/Update Screens.....	2
Browse Screens	3
Standard PF Key Designations	3
Standard Application Screen Messages.....	4
Basic Mapping Support Coding Conventions	4
Identification Division.....	6
Identification Information	6
Area Information	6
Program Function	6
Definition of Files	7
Definition of Tables.....	8
Log of Changes	9
Program Disclaimer	9
Environment Division	9
Cataloged Record and DB2 DCLGEN Description.....	10
Standard Data Names	11
Record Length	12
Picture Clause	12
Value Clause	12
Level Numbers	12
Numeric Fields.....	12
File Description.....	12

Record-Name	13
Redefines Clause	13
Working-Storage Section	13
Linkage Section	15
Procedure Division	16
Modularized Procedure Division.....	16
Mainline Routine	16
Processing Routines.....	16
Input-Output Routines.....	16
Program Size.....	18
Number of VSAM File/DB2 Tables I/Os Within A Transaction	18
Paragraph Names	18
Go To Statements	18
Nested Statements.....	19
IF Statement.....	20
Checking Conditions	20
Perform Statement	20
Move Statement	20
Special Consideration for Selective COBOL Features	21
Date and Time	21
In Working Storage	21
In Procedures Division	21
Standard On-line Field Editing	22
Recommended EXEC CICS Commands And Options.....	23
Error Handling.....	24
Program Control.....	25
File Control.....	26
Transient Data Control.....	29
Terminal Control	29
Interval Control.....	30
Documentation	31
Numeric Deedit of Edited Fields.....	32
Abend Messages and Codes	33

Acknowledgement

Any organization interested in reproducing the COBOL report and specifications (of the Conference on Data Systems Language) in whole or in part, using ideas taken from this report as the basis for an instructional manual or for any other purpose, is free to do so. However, all such organizations are requested to reproduce this section as part of the introduction to their document. Those using a short passage, as in a book review, are requested to mention COBOL in acknowledgment of the source, but need not quote this entire section.

COBOL is an industry language and is not the property of any company or group of companies, or of any organization or group of organizations.

No warranty, expressed or implied, is made by any contributor or by the COBOL committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or by the committee, in connection therewith.

Procedures have been established for the maintenance of COBOL. Inquiries concerning the procedures for proposing changes should be directed to the Executive Committee of the Conference on Data Systems Languages.

The author and copyright holders of the copyright material used herein, Flowmatic™, Programming for the UNIVAC® I and II, Data Automation Systems copyright © 1958, 1959 Sperry Rand Corporation, IBM Commercial Translator Form No. F28-8013, copyright © 1959 IBM, Fact, DA127A5260-2760, copyright © 1960 Minneapolis-Honeywell, have specifically authorized the use of their material in whole or in part in the COBOL specifications. Such authorization extends to the reproduction and use of COBOL specifications in programming manuals or similar publications.

Introduction

Use these standards and guidelines when coding CICS Execute Level COBOL Applications. These standards assume that the reader/user is proficient in the COBOL language.

CICS COBOL II programs contain CICS commands that are translated into COBOL statements that are then compiled in the usual way. When these programs are executed, the statements inserted by the translator invoke the CICS Execute Interface Program (DFHEIP). DFHEIP provides the service that each command requests by invoking one or more CICS control program(s).

In addition to invoking CICS control programs, the Execute Interface program obtains and provides addressability to any required storage areas, such as terminal I/O areas, and to various work areas that are then released automatically when no longer needed. Usually, the application programmer need only select the required function and then code the appropriate command without having to know about CICS storage areas and control blocks. When access to such areas is necessary, the Command-Level provides the ADDRESS command for this purpose as described in the *IBM CICS Command-Level Programmer Reference Manual*.

Rationale for Standards

Computer procedures must be developed and implemented using well-defined standards. Any standard that is used should be designed and implemented to give the highest possible assurance that the system will continue to serve an entity in spite of personnel changeover, changes required in the system to answer the entity's needs, computer changes, etc., at minimum costs.

This handbook deals with the part of computerized procedures known as computer programs. Many computer programs have been developed in what can be termed an undisciplined or ad hoc fashion. Generally, individuals working in computer installations each create and implement these procedures, with good intentions, along individualized and personalized lines that they alone understand. This type of development seldom provides good long-term assurance of system operation at minimum costs. Instead, it usually means that an effort has to be made to reorganize programs when changes occur in personnel, computer configurations, or application systems. Also, this type of development generally will require a larger staff to support programming efforts than would be required if good programming guidelines had been used when the system was developed.

Both beginning and experienced programmers should be interested in this handbook because it provides each with standards by which to develop well-documented programs that are easier to implement and maintain. Also, it should interest data processing managers, programming managers, and others responsible for data processing efforts. For them, it can provide a basis for comparison or suggest ways of developing their own standards. It can also be used as a handbook by their programming staff. For instructors, it should be valuable as an aid in establishing student self-discipline when writing programs.

The Administrative Data Processing Center at Iowa State University has fully implemented the standards set forth in this handbook. Although there is obvious reluctance to change from individual methods on the part of everyone involved in computer program development, experience indicates that practical standards can win the overwhelming approval and support of every true professional involved in the programming effort. Such has been the case at Iowa State University Administrative Technology Services.

The Application Programming Standards Group is responsible for maintaining these standards. The group's objective is to create standards that ensure ease of programming and maintenance, and that provide continuity throughout the Center at the same time.

To ensure that the standards are followed, the committee has established a functioning review process in which the programming staff is actively involved. Each program is thoroughly reviewed, benefiting both the reviewer and the programmer by keeping them abreast of new standards and of changes to old ones.

Purpose of This Handbook

The intent of this handbook is to establish good program development standards for the COBOL language. It is neither a handbook that is all-inclusive as far as standards are concerned; nor is each individual entry closed to debate. However, the manual does provide a good base for further development within a given enterprise that uses the COBOL language.

COBOL can be an adequate language to use for programming computers to process business type data. However, it is a flexible language and is easily misused (like all languages) by programmers. Beginning programmers tend to misuse COBOL because they have not learned the consequences thereof. Experienced COBOL programmers tend to set their own guidelines, which are understood only by them, making them unacceptable as good shop standards.

The standards presented here provide guidelines to write programs that are easily read and maintained. A readable program is one in which the logic and flow of control can be easily discerned. An easily maintained program is one that can be modified without restructuring the program logic. One significant feature of these standards is the use of a modular, top-down approach that uses a restricted form of branches.

The goal of these standards is to ensure that COBOL source programs written according to the standards will:

- Be organized and formatted to enhance readability.
- Document what is being done to solve the problem.
- Produce machine code that is well organized.
- Solve the problem consistently and be able to handle any exceptional situation.
- Sufficiently control the problem so that good testing procedures can determine if the problem to be solved is truly solved when the program is finished.
- Require minimal study and change in order to be maintained.

During the development of these standards, consideration was given toward making them easy to follow. For these standards to be effective they must be followed in all new programs written. Whenever possible, they should be incorporated into modifications of programs that were not originally written according to these standards.

To be effective, the Standards committee needs input from the entire staff. Bring changes and suggestions about the standards to the attention of any committee member.

Testing Procedures and Requirements

One of the ramifications of an interactive on-line environment is that Administrative Technology Services problems are highly visible to users of this service. Therefore, it is extremely important for the system to be up continuously if we are to provide good service. Not all down situations can be prevented, for example, power failures and equipment malfunctions; however, we can prevent most down-time that occurs due to program malfunctions. The important thing to remember is that a malfunction may not only affect the user of the given program but it may also affect all terminal users if it brings CICS down. To prevent CICS from going down, we require that the CICS systems programmer review all new CICS programs and all changes to active CICS programs before they are put into production. Follow the procedures listed in Sections A and B on the next page.

Pre-Production Procedures

1. Review the design and flow of the system.
2. Have data names reviewed and copybooks finalized for new record descriptions.
3. Add file definitions to the FILE DEFINITION system.
4. Compile on-line programs and maps.
5. Create necessary procedures and test data sets.
6. Provide the CICS systems programmer with the information for the necessary CICS control table entries. The information includes map names, program names, and CICS data set names with their associated DSNs. Provide these as the individual programs are developed. Provide transaction IDs for routing-module programs. Do not begin new CICS transaction IDs with the letter *C* because the CICS system has reserved this for its own use.
7. Develop a user manual describing the operation of the on-line system.

Production Implementation Procedures

1. Keep the CICS systems programmer posted concerning the implementation target date.
2. Have all programs go through the program review procedure before being included in the production version. Please submit your programs for review at least one week ahead of your planned implementation date, and earlier if possible.
3. There are separate CICS control tables for test and production versions of CICS. Therefore, you need to provide the CICS systems programmer with a complete list of necessary table entries for the application to be put into production. Do this at least two days before your implementation target date.
4. Because of the high visibility of interactive applications and because of the importance of first impressions, a system must have been brought up in a Pre-Production Test mode so that the analyst in charge can check out the system before the actual implementation of a new or expanded interactive application. When going production, the production system must be tried first, do this later in the evening on second shift, or early on Saturday morning. Also, coordinate this with the CICS systems programmer.

Sign-On and Routing Approach to CICS Application Development

Separate routing and menu module programs are a required part of every on-line application system. The menu program receives control when the system is entered.

The routing module is activated each time the [Enter] key (or a PF key) is pressed. The routing module determines which program should be executed next and transfers control to it.

Sign-On

All new systems will use the ADIN sign on.

The ADIN sign-on program will transfer control to the MENU program of the system and will pass control data via the common area as defined by Copybook WSCAADIN.

The MENU Program

The MENU program will have routines to do the following:

1. Perform additional security and/or access checks, if required.
2. Receive control data from the ADIN sign-on.
3. Display the menu map for the system by building the menu from the MENUssss copybook. The ssss must represent the system.
4. Display any error messages based on the value of a transfer switch.
5. Transfer control back to the ADIN sign-on based on a transfer switch and pass control data back to the ADIN sign-on.

Standard Screen Formats

This section describes standards and guidelines for the screen definition process. Because a large number of our interactive systems are being used by more than one office, the various screens must have a similar design. This will make using the different systems easier. Each user will be familiar with a basic screen layout and will not need to be reeducated about each system.

In these standards, the term **ADIN Menu** applies to a screen that lists the interactive systems available to a particular client. The term **System Menu** applies to a screen that lists the functions available within a particular interactive application system.

Center the following acknowledgment on a separate line under the menu title line of every menu using the normal intensity attribute: DEVELOPED BY IOWA STATE UNIVERSITY ADMINISTRATIVE DATA PROCESSING CENTER. This standard is retroactive for application systems that are distributed to outside sources.

General Guidelines for Screen Definition

1. The first field on each screen must be the FUNCTION field. Locate this field in the upper left corner of the screen. This is normally a two-position field; it may be a three-position field if necessary to satisfy a user's needs. Obtain your team leader's approval before using a three-position field.
2. Avoid cramming as many fields as possible on one screen simply to minimize the number of screens. Overcrowding results in screens that are difficult to use and maintain.
3. If a screen is used for data entry from an input form, design the screen and the form together so they look the same.
4. Design a screen to address the operator directly.
5. Avoid personifying the computer. For example, do not use WELCOME TO THE ABC SYSTEM. Instead just use ABC SYSTEM.
6. Above all, be consistent.

System Menu Screens for Function Selection

1. Use the first line of the System Menu Screen for key fields; two lines may be necessary in some cases. Put headings on all key fields. The sequence of the key fields will depend on the application.
2. Following the key fields, center the title in the top portion of the screen.
3. In the center portion of the screen, list the available functions preceded by their respective function codes.
4. Use line 23 for standard action/error messages.

Display/Add/Update Screens

1. Use the first line of the Display/Add/Update Screens for key fields; two lines may be necessary in some applications.
2. Use the center portion of the screen to display the data.
3. Use line 23 for standard action/error messages.

Browse Screens

1. Use the first line of a browse screen for key fields; two lines may be necessary in some applications. Allow the key used to start the browse to pass along from screen to screen during the browse. Use the [Enter] key to restart the browse.
2. Use the center portion of the screen to display no more than fifteen records.
3. Have each line of the browse start with a field that the operator could use to select a particular record from the browse screen by entering a function code. If the operator enters a function code in this field, control will go through the routing module to the browse program, then back through the routing module to the particular display program depending on the function code that was entered.
4. Use line 23 for standard action/error messages.

Standard PF Key Designations

Program Function key usage should follow the SAA Standards to help standardize PF key definitions.

1. PF1 = Help (when available).
2. PF2 = Verify.
3. PF3 = Exit.
4. PF4 = Transfer to a different system menu.
5. PF5 = Back (by screen).
6. PF6 = Forward (by screen).
7. PF7 = Scroll up.
8. PF8 = Scroll down.
9. PF9 = Scroll left.
10. PF10 = Scroll right.
11. PF11 = Previous record/account.
12. PF12 = Next record/account.

The following display should be on line 24:

-----1-----	-----2-----	-----3-----	-----4-----	-----5-----	-----6-----	-----7-----	-----8
F1=Hlp	F3=Ext	F4=Trn	F5=BSc	F6=FSc	F7=Up	F8=Dn	F9=Lf
							F10=Rt
							F11=Prv
							F12=Nxt
-----1-----	-----2-----	-----3-----	-----4-----	-----5-----	-----6-----	-----7-----	-----8
white		yellow		green		blue	

Standard Application Screen Messages

Use standard operator prompt messages on all screens. Reserve line 23 for screen identification and for standard error/action messages. Make the first field on line 23 five bytes long, and have it contain the program ID used to display the screen. Make the second field 36 bytes long; name it MnnMSGL, and use it to provide abnormal condition error messages like edit errors, file errors, and processing errors. Make the third field 36 bytes long also, but name it MnnMSGR. Use it for displaying normal processing messages like indicating what action was just taken.

1. Abnormal processing messages; MnnMSGL
 - A. EDIT ERROR DETECTED - Errors have been found in the information that was entered. Correct any bright fields before pressing the enter key.
 - B. RECORD NOT FOUND - The requested record was not found.
 - C. INVALID FUNCTION CODE - The entered function code is not available.
 - D. INVALID PF KEY USED - A PF key not defined for this system was used.
 - E. FILE IS NOT OPEN; CONTACT ATS - The user file is not open.
2. Normal processing messages; MnnMSGR
 - A. RECORD MAY BE ADDED - The requested record does not exist but may be added.
 - B. RECORD MAY BE UPDATED - The record displayed on the screen may be updated.
 - C. RECORD HAS BEEN ADDED - The record has been added to the system.
 - D. RECORD HAS BEEN UPDATED - The record has been updated.
 - E. RECORD HAS BEEN DELETED - The record has been deleted.
 - F. PRESS ENTER TO DELETE THIS RECORD - This message verifies a delete request.
3. Other messages besides those mentioned above will be necessary in the interactive systems. The analyst in charge of each system will be responsible for developing additional messages using the standard messages as examples. If necessary, use lines 21 and 22 in addition to line 23 in order to keep all messages in the same area on the screens.

Basic Mapping Support Coding Conventions

1. Code one parameter per line.
2. Begin continuation lines in column 16.
3. Use the (line, column) option to specify field position.
4. Do not allow the value of the LENGTH parameter plus the value of 'column' in the POS parameter to be greater than 80.
5. Use a prefix of Mnn where nn are the last two digits of the map name.

ASKIP:	Specifies that data cannot be keyed into the field and causes the cursor (current location pointer) to skip over the field.
ATTRB:	Is applicable only to fields to be displayed on a 3270 (it is ignored if sent to a non-3270 terminal) and specifies device dependent characteristics and attributes, such as the capability of a field to receive data or the intensity to be used when the field is output.
BRT:	Specifies that a high intensity display of the field is required. By virtue of the 3270 attribute character bit assignments, a field specified as BRT is also potentially detectable. However, for the field to be recognized as detectable by BMS, DET must also be specified.
COLOR:	<p>Indicated the individual color, or the default color for the map set (where applicable). This is overridden by the COLOR operand of the DFHMDI macro, which is in turn overridden by the COLOR operand of the DFHMDF macro.</p> <p>The valid colors are blue, red, pink, green, turquoise, yellow, and neutral.</p>
DRK:	Specifies that the field is nonprint/nondisplay. DRK cannot be specified if DET is specified.
END OF MESSAGE:	Is only used with printer settings.
FSET:	<p>Specifies that the modified data tag (MDT) for this field should be set when the field is sent to a terminal.</p> <p>Specification of FSET causes the 3270 to treat the field as though it has been modified. On a subsequent read from the terminal, this field is read, whether or not it has been modified. The MDT remains set until the field is rewritten without ATTRB=FSET or until an output mapping request causes the MDT to be reset.</p>
NEWLINE:	Is only used with printer settings.
PROT:	<p>Specifies that data cannot be keyed into the field.</p> <p>If data is to be copied from one device to another attached to the same 3270 control unit, the first position (address 0) in the buffer of the device to be copied from must not contain an attribute byte for a protected field. When preparing maps for 3270s, ensure that the first map of any page does not contain a protected field starting at position 0.</p>
UNPROT:	Specifies that data can be keyed into the field.

Identification Division

The identification division contains information about the program and its purpose. An informative identification division can enhance the program's documentation. Accurate information here can save hours of maintenance time and possibly eliminate the need for a new program and/or some external documentation.

All identification division information, except for PROGRAM-ID, must contain an asterisk, '*', in column 7 of the source code when included in a program.

Identification Information

```
-----1-----2-----3-----4-----5-----6-----7--
  IDENTIFICATION DIVISION.
  *
  PROGRAM-ID.      Five position program name
  *PROGRAM-TITLE.  Title of program on Source Control System
  *AUTHOR.         Full name of programmer
  *DATE-WRITTEN.   Date program began
  *
```

Area Information

```
-----1-----2-----3-----4-----5-----6-----7--
  *      *      A R E A   I N F O R M A T I O N      *
  *
  *
  *ANALYST - Full name of analyst in charge of program
  *USER    - User department for which program is being written for
  *SYSTEM  - Full name of the system to which the program belongs
  -----1-----2-----3-----4-----5-----6-----7--
```

Program Function

This is a brief statement of the objectives of the program. Comments at the paragraph level should explain important details not mentioned in the Program Function section.

Definition of Files

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--
*      * * D E F I N I T I O N   O F   F I L E S * *
*
*FILE-ID  FILE-NAME          |      BOOKNAME  I-O
*=====  =====          |      =====  ==
*PR60100  PRMS-PRNT-MAIL-SBSC  PRMSTERB  I-O
*PR60110  PRINTING MAILING MAGAZI PRMAGZNB  I-O
*PR60120  PRINTING MAILING CATEGY PRCATGYB  I-O
*UD60020  USER INPUT DATA - ZIPS  UDINPUTC  I
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--

```

FILE-ID: For permanent files - the file library ID.
 For nonpermanent files - 'WK DISK'
 For sort files - 'SORT WK'

FILE-NAME: A four character prefix followed by descriptive names.
 Should not include the suffix '-FILE'.

BOOKNAME: Source book name(s) used for the file.

I-O: File usage. Input = '_I_' Output = '_O_'
 Input and Output = 'I-O'

Definition of Tables

1	2	3	4	5	6	7
* * * D E F I N I T I O N O F T A B L E S * *						
						JOINED
*FILE-ID	FILE-NAME	BOOKNAME	I-O	SEQ	INDEX	TO TBL
*=====	=====	=====	==	==	=====	=====
*PMB0100	PROJECT MANAGEMENT MAST	PMMSTERD	I	ASC	PMB0103	_____
*PMB0110	PROJECT MEMBERS	PMMEMBRD	I	_____	_____	_____
*ADB0000	CLIENT ID MASTER	ADCMSTRD	I	_____	_____	_____

TBL-ID: The file library ID.

TBL-NAME: A four character prefix followed by descriptive name.

BOOKNAME: Source book name(s) used for the table.

I-O: File usage. Input = '_I_' Output = '_O_'
Input and Output = 'I-O'

SEQ: ASC for ascending, DSC for descending.

INDEX: The file library ID (last position must be 1 through 9).

JOINED
TO TBL: This field is used to join two or more DB2 tables.

The first DB2 table in the join is referred to here as the base table. It should be listed first and the JOINED TO TBL field should be blank (or underscores).

The DB2 tables to be joined to the base table should be listed next. For these tables, only the TBL-ID and JOINED TO TBL fields are used. The base table ID should be entered in the JOINED TO TBL field.

Sample: join tables AAB0100, AAB0110, AAB0120

1	2	3	4	5	6	7
						JOINED
*TBL-ID	TBL-NAME	BOOKNAME	I-O	SEQ	INDEX	TO TBL
*=====	=====	=====	==	==	=====	=====
*AAB0100	AAJN-xxx..	_____	I	ASC	AAB0101	_____
*AAB0110	_____	_____	_____	_____	_____	AAB0100
*AAB0120	_____	_____	_____	_____	_____	AAB0100

Log of Changes

All program changes must be recorded in the Source Control System when a program is being checked out or checked back into production. Programmers are no longer required to make an entry in the Log of Changes section of the program when a program is modified. If a programmer wishes to maintain the log, all entries would contain the following information:

DATE: Date code was first changed.

NAME: Full name of person responsible for the change.

DESCRIPTION: Explanation of the change.

Program Disclaimer

For programs leaving the department, the following statements are required at the beginning of the Identification Division.

Usage - All rights reserved. The recipient may use for non-profit purposes only and may distribute only with written approval from Iowa State University - Administrative Technology Services.

Disclaimer - Although tested, no warranty, expressed or implied, is given to this program by Iowa State University.

Environment Division

The environment division is unnecessary for CICS COBOL. The CICS systems programmer maintains the information necessary to describe a VSAM file in the CICS file control table.

Data Division

The data division should describe all the data that the program uses. These standards assume extensive use of cataloged record and DB2 DCLGEN descriptions, which leads to two major benefits. First and most importantly, it ensures that record and DB2 DCLGEN descriptions and data names will be identical in all programs using the same file. Secondly, it eliminates duplicate coding of the record or DB2 DCLGEN description in each program.

Cataloged Record and DB2 DCLGEN Description

1. All new cataloged record and DB2 DCLGEN descriptions must be developed using the ATS Data Dictionary System.
2. The analyst is strongly encouraged to move old cataloged record descriptions from COB1.PROD.COPYLIB to the Data Dictionary System. Contact the Data Dictionary Coordinator for assistance.
3. Catalog all DB2 DCLGEN and record description if used in more than one program or if there is an active tape or disk file in our library.
4. Access a cataloged record description by using a COPY statement. Always begin the reserved word "COPY" in column 12. If a host variable from a registered cataloged record description is used in an SQL statement, the record description will need to use INCLUDE instead of COPY.
5. Access a cataloged DB2 DCLGEN description by using an SQL INCLUDE statement. Always begin the reserved word INCLUDE of the EXEC SQL INCLUDE command in column 16.
6. The analyst in charge of the file will maintain the cataloged record or DB2 DCLGEN description.
7. Refer to the ATS Data Dictionary System Manual for a complete discussion of data naming conventions. The cataloged record or DB2 DCLGEN description name shall conform to the following:
 - a. The name shall be eight characters long.
 - b. The first two characters shall be the area prefix. New prefixes may be assigned by the team leaders.
 - c. The second two characters shall uniquely identify the file within an area.
 - d. The eighth character shall be a 'C' for Cobol record and a 'D' for DB2 DCLGEN descriptions. Easytrieve Plus (E) record descriptions are generated from the COBOL record description.

```

-----1-----2-----3-----4-----5-----6-----7--
*RECORD LAYOUT:  SYSTEM NAME=Inst. Research Information System
*                FILE NAME=IRES DEPT ORG TBL BACKUP
*                SKIP PREFIX=YES
*                RECORD NAME=IRES DEPT ORG REC
*                RECORD LENGTH= 49
*                CREATE DATE=05/08/91
*                ATS EMPLOYEE NUM= 31 TLB
*
01  IRDE-IRES-DEPT-ORG-REC.
    12  IRDE-IRES-SNPSHT-DATE-DB.
        16  IRDE-IRES-SNPSHT-CCYY.
            20  IRDE-IRES-SNPSHT-CC          PIC X(2).
            20  IRDE-IRES-SNPSHT-YY          PIC X(2).
        16  IRDE-IRES-SNPSHT-DASH1          PIC X(1).
        16  IRDE-IRES-SNPSHT-MM             PIC X(2).
        16  IRDE-IRES-SNPSHT-DASH2          PIC X(1).
        16  IRDE-IRES-SNPSHT-DD             PIC X(2).
    12  IRDE-IRES-ORG-GRP-NUM                PIC X(5).
    12  IRDE-IRES-DEPT-NUM                  PIC X(5).
    12  IRDE-IRES-DEPT-SHORT-NAME            PIC X(15).
    12  IRDE-IRES-DEPT-ABRVN                 PIC X(5).
    12  IRDE-IRES-XDSCPLN-DEPT-NUM          PIC X(5).
    12  IRDE-IRES-DEPT-CTGRY                PIC X(1).
        88  IRDE-IRES-ACDMC                  VALUE 'A'.
        88  IRDE-IRES-ADMNV                  VALUE 'M'.
        88  IRDE-IRES-ADVSG                  VALUE 'V'.
        88  IRDE-IRES-RSRCH                  VALUE 'R'.
        88  IRDE-IRES-DCMTN                  VALUE 'O'.
        88  IRDE-IRES-DEAN-OFFICE             VALUE 'D'.
    12  IRDE-IRES-INCLD-STAFF                PIC X(1).
    12  IRDE-IRES-INCLD-STDNT                PIC X(1).
    12  IRDE-IRES-INCLD-CRSE                 PIC X(1).
    12  IRDE-IRES-INCLD-FNCL                 PIC X(1).
    12  IRDE-IRES-INCLD-BLDG                 PIC X(1).
-----1-----2-----3-----4-----5-----6-----7--

```

8. New cataloged record and DB2 DCLGEN descriptions must be reviewed and approved before use in production programs. When using the 'PE' function of the Data Dictionary System, the descriptions will be listed along with the check-off review sheet. Send this to the first person on the check-off sheet.

Standard Data Names

1. Use standard, meaningful data names. This implies that anyone familiar with the application area will be able to read the program and know what each data name represents.
2. The following are examples of data names: 'IRDE-IRES-DEPT-NUM' would describe the university department number within the Institutional Research department organization file. C1, P1, BO, SO, C-X-Z, etc. are examples of meaningless names that should not be used.
3. Refer to the ATS Data Dictionary System Manual for further examples of standards for commonly used data names.
4. Condition Names should be set up for items with defined values. Condition names for a data item will be generated as 88 level entries.

Record Length

The standard print record length is 133 characters. Use the first character for line/page spacing control. (1 = new page, blank = single space, 0 = double space, - = triple space)

Picture Clause

1. Vertically align picture clauses in the program.
2. To fit the statement on one line, the PIC clause may be shifted left or right.
3. Use the abbreviation PIC.

Value Clause

1. Vertically align VALUE clauses in the program.
2. To fit the statement on one line, the VALUE clause may be shifted left or right.

Level Numbers

1. Always stagger level numbers; put the smallest number to the left, and move at least 2 columns to the right for each higher level number.
2. Use noncontiguous level numbers such as '12', '16' and '20'. Use a column number such as 12 if the 12 begins in column 12.

Numeric Fields

1. Always sign quantitative numeric fields.
2. When at all possible, make all numeric fields COMP or COMP-3.
3. Use an odd number of digits for COMP-3 field definitions.
4. Use the abbreviations COMP, and COMP-3.

File Description

1. Describe files in the same sequence as they are selected.
2. Start all file names in column 12 and all FD clauses in column 16.
 - a. Begin File-Description (FD) names with a prefix of four or more characters and end with '-FILE'.
 - b. Make the record name at the 01 level in an FD the FD name ending with '-REC' instead of '-FILE'.
 - c. Begin all levels in the FD record with the FD name prefix.
3. Use the following FD clauses where applicable:

- a. RECORDING MODE IS....
- b. BLOCK CONTAINS....
Use block contains 0 records in the FD for all non-VSAM files.
- c. RECORD CONTAINS....

Record-Name

Always use a meaningful record name that identifies the record being defined.

Redefines Clause

1. Always make redefines the first clause of a redefining statement.
2. Example: 01 DATA-NAME-1 PIC XX.
 01 DATA-NAME-2 REDEFINES DATA-NAME-1 PIC S99.
3. If a redefines statement is used in an SQL statement all fields in the redefines structure must be named other than FILLER.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--
      01  WSWA-AREAS.
          12  WSWA-ALUM-NUMBER          PIC X(9)          VALUE SPACES.
          12  WSWA-ALUM-NUM REDEFINES  WSWA-ALUM-NUMBER.
              16  WSWA-ALUM-NUM-1-7     PIC X(7) .
              16  WSWA-ALUM-NUM-8      PIC X.
              16  WSWA-ALUM-NUM-9      PIC X.
      *
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--

```

The data division for a CICS COBOL program consists of two sections: working-storage and linkage. The file section is not used. The CICS systems programmer maintains the information necessary to describe a file in the CICS file control table.

Working-Storage Section

The working-storage section of a CICS COBOL program will contain all necessary constants, data elements, key fields, cataloged record descriptions, map definitions, and copied system support modules.

The accepted method of coding the working-storage section will depend entirely upon the on-line system. However, maintain consistency between programs within a system. The following is a suggested order for coding the working-storage section.

1. Data names in the working storage section should begin with a four character prefix and the first two should be 'WS'.
Example: WSWA-, WSPH-
2. Give initial values to Working-Storage items.
3. In order to facilitate debugging from program dumps, make the first entry in the working-storage section the literal 'PPPPP WS BEGINS' (PPPPP is the program number). Add other entries to facilitate debugging.
4. Code all constants at the start of working-storage.
5. Code all file-key data elements.
6. When temporary storage records are used in a system, use the following as the name of the temporary storage queue:

```
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--
      12  WS-QUEUE-NAME.
          16  WS-QUEUE-TERMINAL-ID      PIC X(4)
          16  WS-QUEUE-TRANSACTION-ID   PIC X(4)
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--
Temporary Storage Queue
```

7. Code all variable data elements. Use the COMMON-Area to pass control fields between programs. This will be a source book named WSCAssss where ssss is the system code. The following describes the usage of the WSCA-TRANSFER-SW field:
 - A. Set the value of this field.
 - B. Use the length of the common area to be passed in the LENGTH parameter (LENGTH = WSCAssss), where ssss is the system number.
 - C. Execute an XCTL, LINK, or RETURN command.For example, the routing program would set this switch to a 2, the common area length to 900 and then transfer control (XCTL) to the router when an individual record is selected the browse screen.
For example, any program that returns control to CICS by the RETURN command would set this switch to an R and the common length to 900 before executing the RETURN command.
8. Code all copied BMS DSECTS.
9. Code all copied system support modules. Code only those required for processing in each program.
 - A. DFHBMSCA - Standard attribute list. This book may be copied into programs to obtain a set of commonly used IBM 3270 terminal field attributes.
 - B. DFHAID - Standard attention identifier list. This book may be copied into programs to obtain a set of 3270 attention identifiers that can be used to test the value of EIBAID.
10. Code all copied file copybooks.

Linkage Section

Use the linkage section to define common areas that were passed from some other program. The most common use of this is the passing of data from the routing module to subordinate programs. The data elements that are to be passed will be defined in a copybook (WSCAXXXX) that will be copied into working storage. A parameter of the XCTL command will point to this 01 level. Do not change data in the linkage section; move the data to the working storage area and make changes there. Use the following guidelines in programs that require information from the common area.

1. The first statement coded in the linkage section must be 01 DFHCOMMAREA.
2. All data elements in the DFHCOMMAREA must be on a level lower than 01. Failure to do this will cause unpredictable results.
3. Do not modify data in the LINKAGE Section. (Don't, for example, code any MOVE TOs the Linkage Section.)
4. The following are examples of linkage sections by program type:

```

-----1-----2-----3-----4-----5-----6-----7--
LINKAGE SECTION.
01 DFHCOMMAREA.
   12 LSCA-FROM-ROUTING.
      16 LSCA-TRANSFER-SW          PIC X.
      16 FILLER                     PIC X(1899).
   12 FILLER REDEFINES LSCA-FROM-ROUTING.
      16 LSCA-FROM-ADIN             PIC X(700).
      16 FILLER                     PIC X(1200).
-----1-----2-----3-----4-----5-----6-----7--

```

Menu Program

```

-----1-----2-----3-----4-----5-----6-----7--
LINKAGE SECTION.
01 DFHCOMMAREA.                                PIC X (900)
-----1-----2-----3-----4-----5-----6-----7--

```

Routing Program

```

-----1-----2-----3-----4-----5-----6-----7--
LINKAGE SECTION.
01 DFHCOMMAREA.
   12 FILLER                     PIC X (900).
   12 LSCA-INPUT-MESSAGE          PIC X.
-----1-----2-----3-----4-----5-----6-----7--

```

Other CICS Programs

Procedure Division

Each application program should be single transaction oriented to optimize response time and minimize main storage requirements. Each program should perform only one function such as displaying a record for update.

Consider the following list of procedure division standards while coding or modifying a CICS application program.

Modularized Procedure Division

These standards organize the procedure division into three modules, mainline, processing, and input-output. The mainline module contains the statements for all the major logical decisions in the program. The mainline decisions drive the data manipulation subroutines contained in the processing module. The input-output module contains the file processing subroutines. All subroutines are closed; that is, they have only one entry point and one exit point.

Modularization will involve subdividing the procedure division into at least three functional levels: mainline routine, processing routines, and input-output routines. This type of structure will allow for ease of maintenance and documentation. Separate each functional level with headings as in the following:

Mainline-Routine

Processing-Routines

Input-Output-Routines

Mainline Routine

Make the mainline routine the controlling module of the program; have it determine and control the order in which processing routines are executed, and give it access to all other routines contained in the program. Use PERFORM statements to invoke processing routines from the mainline routine. In the mainline routine, manipulate data necessary for a mainline decision. Do all other data manipulation and table handling in the processing routines.

Processing Routines

Break down processing routines into as many functional levels as necessary, depending on the complexity of the program. Make these completely closed routines, with only one entry point and only one exit point. Make the exit point the exit paragraph of the paragraph being performed. Allow processing routines to perform only input-output routines and other processing routines according to the rules governing the PERFORM statement. Have all processing and manipulating of data done in the processing routines before performing input-output routines.

Input-Output Routines

Make the input-output routines the lowest level routines, (910 - 989) since all higher level (100 - 849) routines will have access to them. Programs should contain only one of each form

of the I/O statements that are valid for each file, for example, READ, WRITE, REWRITE, etc. This may mean moving the data to a common area before issuing a write.

Program Size

If the total size of the object programs within a transaction exceeds 32k plus file record description plus BMS map lengths, the transaction will be placed in a lower priority class.

Number of VSAM File/DB2 Tables I/Os Within A Transaction

If the total number of file I/Os (including common activity and journaling puts) exceeds 20 in the display/update program, 30 for VSAM, and 40 for DB2 in the browse programs, the transaction may be placed in a lower priority class. Also include in the count the I/O required by the system to maintain an alternate index.

Paragraph Names

Each paragraph name must consist of a three-digit ascending prefix followed by a verb-object description.

Examples: 100-CHECK-PASSWORD or 125-READ-DPCNTRL

Go To Statements

There are two styles of CICS programs, old and new. An old style program is written in COBOL and converted to COBOL II. The program starts execution at the top of the procedure division and uses a fall-through approach to reach the end of the program. GO TOs are used frequently in these programs and do not violate ATS standards.

A new style program is one that has been generated using the COBOL II language. This type of program uses a PERFORM approach similar to that used in ATS batch programs. Paragraphs are performed rather than using GO TOs. The only exceptions to this are when the program will exit with either a CICS EXEC RETURN, CICS EXEC XCTL, or CICS EXEC ABEND. The only use of the GO TO statement allowed will be to a paragraph executing one of these statements or to a paragraph exit.

Nested Statements

IF, Inline PERFORM, EVALUATE

1. The most important consideration for nested statements is that the code can be easily understood and maintained by other analysts in the future. Avoid using complex combinations of AND/OR conditions, combinations of multiple nesting levels, or large numbers of statements within the nesting.
2. Statements may be nested to no more than three levels in any combination.
3. Explicit scope delimiters must be used to denote a nested verb's range. The delimiter should be aligned vertically with the verb with which it is associated.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--
      IF ACTIVE-RECORD
        MOVE DATA-NAMAE-1          TO DATA-NAME-2
        IF HEADER-REC
          PERFORM 300-HEADER-RTN          THRU 300-EXIT
        ELSE
          PERFORM 400-HEADER-RTN          THRU 400-EXIT
        END-IF
        PERFORM 500-PROCESS-DETAIL        THRU 500-EXIT
      END-IF

      PERFORM
        VARYING WSTB-TABLE-SUB FROM 1 BY 1
          UNTIL WSTB-TABLE-SUB > WSTB-TABLE-MAX
            OR WSTB-FIELD-1 = SPACES
        IF WSTB-FIELD-2 (WSTB-TABLE-SUB) > 500.00
          SET WSTB-FIELD-3 (WSTB-TABLE-SUB) TO 'Y'
        ELSE
          SET WSTB-FIELD-3 (WSTB-TABLE-SUB) TO 'N'
        END-IF
      END-PERFORM

      EVALUATE TRUE
        WHEN WSPR-SINGLE-SPACE
          ADD +01          TO WSRC-LINE-COUNT
        WHEN WSPR-DOUBLE-SPACE
          ADD +02          TO WSRC-LINE-COUNT
        WHEN WSPR-TRIPLE-SPACE
          ADD +03          TO WSRC-LINE-COUNT
      END-EVALUATE
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--

```

IF Statement

1. Use parentheses to clarify the sequence of operations and the readability of compound conditions.

```
-----1-----2-----3-----4-----5-----6-----7--
      IF (DATA-NAME-1 > DATA-NAME-5
          AND DATA-NAME-3 > DATA-NAME-4 )
          OR (DATA-NAME-5 < DATA-NAME-6
              AND DATA-NAME-7 < DATA-NAME-8 )
          MOVE DATA-NAME-1          TO DATA-NAME-2
          MOVE DATA-NAME-3          TO DATA-NAME-4
          ADD DATA-NAME-5           TO DATA-NAME-6
      END-IF
-----1-----2-----3-----4-----5-----6-----7--
Using Parentheses
```

2. Code only one condition per line; begin subsequent lines with the logical connective.

```
-----1-----2-----3-----4-----5-----6-----7--
      IF A = B
          OR C = D
      END-IF
-----1-----2-----3-----4-----5-----6-----7--
One Condition per Line
```

3. Code the ELSE clause on a line by itself and align it with the 'IF' that corresponds to it.
4. Use an END-IF Scope delimiter with all IF statements.

Checking Conditions

Check the condition occurring most frequently first.

Perform Statement

1. A PERFORM THRU statement must perform through an EXIT statement.
2. Do not use recursive performs.

Move Statement

1. Use only one MOVE statement per line.
2. Align the MOVE verb vertically when several consecutive moves are involved.

Special Consideration for Selective COBOL Features

- The following COBOL verbs are not approved for use in CICS programs because they can bring CICS down:

ACCEPT	OPEN	SORT
ALTER	READ	START
CLOSE	RERUN	STOP
MERGE	REWRITE	WRITE

- The following COBOL verbs are not approved for use because better alternatives are available:

STOP RUN (use EXEC CICS RETURN)
CALL (use EXEC CICS LINK)

Date and Time

Do not use the DATE and TIME registers for date and time. Instead, link to DP997.

In Working Storage

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--
      12  WSWA-LINK-DP997              PIC X(8)          VALUE 'DP997  ' .
      *
      COPY DPDATESC.
      *
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--

```

In Procedures Division

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--
      MOVE SPACES                      TO DPDA-ADP-DATE-FIELDS
      SET  DPDA-ADP-DATE-OPT-LKUP      TO TRUE
      MOVE '00'                       TO DPDA-ADP-CNTL-DATE-CD
      MOVE 'HH:MM:SS'                 TO DPDA-ADP-FRMT-TM-FRMT
      PERFORM 992-LINK-DP997           THRU 992-EXIT
      IF      CICS-RESPONSE-NORMAL
        AND DPDA-ADP-DATE-VLD
        CONTINUE
      ELSE
        MOVE 'INVALID DATE LOOKUP'    TO M10MSGLO
        MOVE 'CONTACT ATS ANALYST'    TO M10MSGRO
        MOVE WSWA-CURSOR-ON           TO M10FUNCL
        GO TO 700-EXIT
      END-IF
      MOVE DPDA-ADP-CLNDR-DATE         TO M10DATEO
      MOVE DPDA-ADP-FRMT-TM           TO M10TIMEO
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--

```

After the link to DP997, date information will be in the copybook DPDATESC. The current time will be in the field DPDA-ADP-FRMT-TM in the format of hh:mm:ssssssss.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--
992-LINK-DP997.
      EXEC CICS LINK
          PROGRAM (WSWA-LINK-DP997)
          COMMAREA (DPDATESC)
          RESP (CICS-RESPONSE)
      END-EXEC
      EXIT.
992-EXIT.
      EXIT
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--

```

Standard On-line Field Editing

1. Normal requirements:

A. Define working storage fields as in the following example:

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--
12 WSWA-CURSOR-ON PIC S9(4) VALUE-1 COMP.
12 WSWA-EDIT-ERROR-SW PIC X VALUE 'N'.
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7--

```

- B. Editing starts with the top-left field and proceeds sequentially to the bottom-right field.
- C. Check that each map field length is > +0 before attempting to edit a field.
- D. If a field is in error, do the following:
 - i. Set the edit-error switch ON.
 - ii. Set the field's attribute to FSET, BRIGHT.
 - iii. Change the field's length byte by moving WSWA-CURSOR-ON to the field's length byte.
- E. After all the fields have been edited, check WSWA-EDIT-ERROR-SW for the status of the edits:

If the status = Y, and the left message field is blank spaces, move EDIT ERROR DETECTED to the left message field. Use the right message field to display more detailed error messages.

If the status = N, move WSWA-CURSOR-ON to the desired field in the key/function line and ensure that all field attributes have been set to ASKIP.
- F. When the map is sent, use the format as prescribed by the SEND explained under the Program Control section in this manual.

2. Results of edit:

- A. The cursor is positioned on the key/function field or on the first field in error.
- B. All fields in error are bright intensity; the remaining fields are normal intensity.
- C. EDIT ERROR DETECTED or a field-specific message is displayed to the left side of line 23 if an error exists.
- D. All fields that passed their edits are displayed FSET at normal intensity.

3. The following example illustrates an edit:

```

-----1-----2-----3-----4-----5-----6-----7--
200-EDIT-FIELD-1.
      IF M99FLL1L = +0
          MOVE XXXX-FILE-FIELD-1      TO M99FLD10
          MOVE DFHBMASK                TO M99FLD1A
          GO TO 200-EXIT
      END-IF
      IF M99FLD1I = 'Y'
          OR = 'N'
          MOVE M99FLD1I                TO XXXX-FILE-FIELD-1
          MOVE DFHBMAF                TO M99FLD1A
      ELSE
          MOVE DFHBMFBS                TO M99FLD1A
          MOVE WSWA-CURSOR-ON          TO M99FLD1L
          MOVE 'Y'                    TO WSWA-EDIT-ERROR-SW
      END-IF
-----1-----2-----3-----4-----5-----6-----7--

```

Recommended EXEC CICS Commands And Options

The following list of CICS exec commands is not meant to be a complete list of all available commands and options. However, this list should provide basic instructional support to code a typical execute level program. If the list of commands does not include an available command in which you are interested, check with the systems programmer because our installation may not support all of the commands.

All CICS Execute Level instructions begin with the code EXEC CICS in column 12 and end with the code END-EXEC. This coding is not illustrated in the following list of commands but must always be used when coding a command in a program. Do not use an EXEC CICS statement within an IF statement. The general format of a CICS command is as follows:

```

-----1-----2-----3-----4-----5-----6-----7--
      EXEC CICS SEND
          MAP      (WSWA-MAP-NAME-CSZ08)
          FROM      (CSZ080)
          CURSOR    (WSWA-CURSOR-POS)
          FREEKB
      END-EXEC
-----1-----2-----3-----4-----5-----6-----7--

```

Error Handling

RESP Parameter - The RESP parm will return a value indicating the results of the CICS command on which it is present. RESP should be used on most file I/O commands, and any other command where the application can expect errors to occur and wants to handle the errors, rather than letting CICS abend the task. The older HANDLE command should not be used, since RESP is more readable, and doesn't use implied GO TOs, as HANDLE does.

The RESP code must be checked after each CICS command that uses it. The copybook CICSRESP contains 88-levels for common responses. See the following example:

```
-----1-----2-----3-----4-----5-----6-----7--
      EXEC CICS READ
          DATASET(WSWA-FILE)
          INTO(JPZ01)
          RIDFLD(WSWA-KEY)
          LENGTH(WSWA-LENGTH)
          RESP(CICS-RESPONSE)
      END-EXEC

      IF CICS-RESPONSE-NORMAL
          CONTINUE
      ELSE
          MOVE 'READ' . . .
          .
          .
          PERFORM 890- . . .
      END-IF
-----1-----2-----3-----4-----5-----6-----7--
```

Linking to Other Programs

Use the CICS LINK command to transfer control to another program expecting return. Use standard datanames in the LINK to allow accurate cross-referencing. For links to fixed program names, use the standard format WSWA-LINK-XXXXX defined with a VALUE "XXXXX" that is the program name. For links to variable program names, use the standard format WSWA-LINK-PROGRAM-ID, and move WSWA-LINK-XXXXX to WSWA-LINK-PROGRAM-ID prior to the link.

The cross-reference system will 1) create an entry for each dataname defined as WSWA-LINK-XXXXX, even if it is not used in the LINK command; 2) list error messages for undetermined links where the errors must be corrected to match the naming standard; and 3) bypass error messages for links using the name WSWA-LINK-PROGRAM-ID.

```
-----1-----2-----3-----4-----5-----6-----7--
      EXEC CICS LINK
          PROGRAM (WSWA-LINK-XXXXX)
          COMMAREA (copybook)
          RESP (CICS-RESPONSE)
      END-EXEC
-----1-----2-----3-----4-----5-----6-----7--
```

Program Control

DUMP	Dumps all task related data areas.	
	TASK	Required
	DUMPCODE('XXXX')	Any 4 characters <i>or</i>
	DUMPCODE(Ws-Dataname)	Where WS-Dataname is X(4)
ABEND	Dumps all task related data areas and terminates the transaction.	
	ABCODE('XXXX')	Any 4 characters <i>or</i>
	ABCODE(Ws-Dataname)	Where Ws-Dataname is X(4)
XCTL	Transfers control to a program without expecting return.	
	PROGRAM(Ws-Dataname)	Required, Ws-Dataname is X(8)
	COMMAREA(Ws-Dataname2)	Required, any level 01 area
LINK	Transfers control to another program expecting return.	
	PROGRAM(WSWA-LINK-XXXXX)	Required, WSWA-LINK-XXXXX is X(8) and specifies the program name in the VALUE clause. If the linked program name can vary, use dataname WSWA-LINK-PROGRAM-ID.
	COMMAREA(copybook)	Required, the level 01 name of the copybook defining the called program interface.
RETURN	Returns control to CICS if the program was invoked via the 'XCTL' command or will return control to the requesting program if this program was invoked via the 'LINK' command.	
	TRANSID(Ws-Dataname)	Required, Ws-Dataname is X(4) and specifies next transaction to execute when a PF or ENTER key is pressed.
	LENGTH(length of DPDATE\$C)	Required

File Control

READ	Reads a record from a direct-access Dataset.	
	DATASET(Ws-Dataname)	Required, Ws-Dataname is X (8) and is the CICS file control table (FCT) name.
	RIDFLD(Ws-Dataname2)	Required, where Ws-Dataname2 is the dataname holding the key in working-storage.
	INTO(File-Desc)	Required, where File-Desc is the level 01 copied source book of the file description.
	LENGTH(Ws-Dataname3)	Required when reading variable length records. Ws-Dataname3 is S9(4) Comp and contains the maximum length of the file record.
	EQUAL	Required when a specific record is desired, specifies only a record with a key equal to the one in Ws-Dataname2 will satisfy the request.
	GTEQ	To be used in place of EQUAL when a record with a key greater than or equal to the one in Ws-Dataname 2 will satisfy the request. Use the STARTBR/READNEXT routines if than one successive record is desired.
	UPDATE	Required when the record read is to be updated.
WRITE	Writes a record to a direct-access Dataset.	
	DATASET(Ws-Dataname)	Required, Ws-Dataname is X(8) and is the CICS FCT name.
	RIDFLD(Ws-Dataname2)	Required, where Ws-Dataname2 is the dataname holding the key in working storage.
	FROM(File-Desc)	Required, where File-Desc is the level 01 copied source book of the file description.
	LENGTH(Ws-Dataname3)	Required, Ws-Dataname3 is S9(4) COMP and contains the length of the file record.

REWRITE	Rewrites a record to a direct-access Dataset that has been previously read with the update option.	
	DATASET(Ws-Dataname)	Required, Ws-Dataname is X(8) and is the CICS FCT name.
	FROM(File-Desc)	Required, where File-Desc is the level 01 copied source book of the file description.
	LENGTH(Ws-Dataname3)	Required, Ws-Dataname3 is S9(4) Comp and contains the length of the file record.
DELETE	Deletes a record from a direct-access Dataset that has been previously read with the update option.	
	DATASET(Ws-Dataname)	Required, Ws-Dataname is X(8) and is the CICS FCT name.
UNLOCK	Releases areas acquired from a direct-access Dataset.	
	DATASET(Ws-Dataname)	Required, Ws-Dataname is X(8) and is the CICS FCT name.
STARTBR	Sets the starting point for a browsing operation.	
	DATASET(Ws-Dataname)	Required, Ws-Dataname is X(8) and is the CICS FCT name.
	RIDFLD(Ws-Dataname2)	Required, where Ws-Dataname2 is the Dataname holding the key (or partial key) in working-storage. (Even if a partial key is used, RIDFLD must be large enough to hold the full key.)
	KEYLENGTH(Ws-Dataname3)	Required, where Ws-Dataname3 is S9(4) COMP and contains the length of the full or partial key used.

READNEXT	Reads the record from a direct-access Dataset logically next on the Dataset.	
	DATASET(Ws-Dataname)	Required, Ws-Dataname is X(8) and is the CICS FCT name.
	RIDFLD(Ws-Dataname2)	Required, where Ws-Dataname2 is the Dataname holding the key in working-storage and used in the STARTBR.
	INTO(File-Desc)	Required, where File-Desc is the level 01 copied source book of the file description.
	LENGTH(Ws-Dataname3)	Required, Ws-Dataname3 is S9(4) COMP and contains the length of the file record.
	KEYLENGTH(Ws-Dataname4)	Required, where Ws-Dataname4 is S9(4) COMP and contains the length of the full or partial key used.
READPREV	Reads a record from a direct-access Dataset sequentially previous on the Dataset.	
	DATASET(Ws-Dataname)	Required, Ws-Dataname is X(8) and is the CICS FCT name.
	RIDFLD(Ws-Dataname2)	Required, where Ws-Dataname2 is the Dataname holding the key in working-storage and used in the STARTBR. (A NOTFND condition will result if this key is not on the file.)
	INTO(File-Desc)	Required, where File-Desc is the level 01 copied source book of the file description.
	LENGTH(Ws-Dataname3)	Required, Ws-Dataname3 is S9(4) COMP and contains the length of the file record.
RESETBR	Resets the starting point of a browse operation, it is equivalent to issuing an ENDBR/STARTBR sequence.	
	DATASET(Ws-Dataname)	Required, Ws-Dataname is X(8) and is the CICS FCT name.
	RIDFLD(Ws-Dataname2)	Required, where Ws-Dataname2 is the Dataname holding the key (or partial key) in working-storage.
	KEYLENGTH(Ws-Dataname3)	Required, where Ws-Dataname3 is S9(4) COMP and contains the length of the full or partial key used.
ENDBR	Terminates a browsing operation and should be used with all browses.	
	DATASET(Ws-Dataname)	Required, Ws-Dataname is X(8) and is the CICS FCT name.

Transient Data Control

WRITEQ	Writes a record to a sequential data disk set such as the common activity file.	
TD	Required.	
QUEUE('XXXX')	Required, where XXXX is the QUEUE(Ws-Dataname1) destination ID, such as CSAF, or where Ws-Dataname1 is PIC X(4) and contains the Destination ID.	
FROM(Record-Descr)	Required, where Record-Descr is the start of the data area that is to be written.	
LENGTH(Ws-Dataname2)	Required, where Ws-Dataname2 is defined as PIC S9(4) COMP and contains the length of the data to be written.	

Terminal Control

RECEIVE	Receives terminal input into a specified area.	
MAP(Ws-Dataname)	Required, if BMS formatting is desired. Ws-Dataname is X(8) and is the copied source DSECT of a map (Ex. SPZ10).	
INTO(Ws-Dataname2)	Required, Ws-Dataname2 is any data-area when the map option is not specified. When the map option is specified, Ws-Dataname2 is the 'I' DSECT of a map (Ex. SPZ10I).	
LENGTH(Ws-Dataname3)	Required, Ws-Dataname3 is S9(4) COMP. It is set to the maximum input message length prior to executing this command when the map option is not used. After successful completion of this command, Ws-Dataname3 can be interrogated for the actual input message length. This length should be passed through the common area of the routing program so that it can be used on subsequent RECEIVE commands when the map option is specified. This will assure BMS of proper field alignment.	
FROM (WSCA-INPUT-MESSAGE)	Linkage section variable.	

SEND	Sends terminal output to a specified area.	
	MAP(Ws-Dataname)	Required, if BMS formatting is desired. Ws-Dataname is X(8) and is the copied source DSECT of a map (Ex. SPZ10).
	FROM(Ws-Dataname2)	Required, Ws-Dataname2 is any data-area when the map option is specified. Ws-Dataname2 is the 'O' DSECT of a map (Ex. SPZ10O).
	LENGTH(Ws-Dataname3)	Required, Ws-Dataname3 is S9(4) COMP. It is set to the maximum output message length prior to executing this command.
	CURSOR	The cursor will be placed on the first map field with a value of -1 in its length attribute field.
	ERASE	Required, unless old data is to remain on the terminal screen
	FREEKB	Required.

Interval Control

START	Starts a task at some specified time and optionally, to pass some data to that task.	
	TRANSID	Required, identifies the transaction-ID in the CICS tables which will point to a program associated with that transaction-ID.
	INTERVAL(0)	Required, determines that the task is to start immediately.
	TERMID(Ws-Dataname1)	Identifies a terminal (printer) to which the task will be attached. Ws-Dataname1 is a four-character terminal identifier.
	FROM(Ws-Dataname2)	allows data to be passed to the initiated task. Ws-Dataname2 identifies the beginning of a data area in working storage that is to be passed.
	LENGTH(Ws-Dataname3)	allow data to be passed to the initiated task. Ws-Dataname3 is S9(4) COMP and contains the length of the data area to be passed.

RETRIEVE	Retrives the area passed to an initiated task from a previous START command.	
	INTO(Ws-Dataname1)	Identifies an area in working-storage where the passed data can be accessed.
	LENGTH(Ws-Dataname2)	Identifies the length of the data area that was passed. Ws-Dataname2 is S9(4) COMP.

Documentation

Use comment lines (an asterisk in column seven), to explain why particular branches are being made within the program and under what conditions control is being given to another program as a result of a LINK or an XCTL.

Numeric Deedit of Edited Fields

A special routine is available to aid in the deedit of numeric fields. Its use is described as comments at the beginning of the copied source code. Two books are involved:

JPEDTNUM- This is a routine that will deedit numeric fields (right justify, zero fill, and sign fields). The routine is valid for fields ranging in size from S9 to S9(12)V9(6). The 18 digits also can be redefined in your program if other formats are needed.

JPEDWSWA- This is a book of the necessary data elements used by the JPEDTNUM routine.
(Copied into working-storage)

Abend Messages and Codes

Transaction dumps obtained from EXEC level applications usually contain one of the following abend codes. These transaction dumps are normally obtained when an exceptional condition has occurred for which no EXEC CICS handle condition is active. A full list of abend codes and messages is available in the *CICS Messages and Codes Manual*. The more frequent abend codes and their associated conditions are identified in the following table.

Abend Code	Exceptional Condition(s)
AEIL	DSIDERR Data set ID cannot be found in the CICS file control table.
AEIM	NOTFND Record not found.
AEIN	DUPREC An attempt was made to add a record to a dataset in which the same key already exists.
AEIO	DUPKEY Occurs if a record is retrieved via an alternate index in which the key that is used is not unique.
AEIP	INVREQ FILE CONTROL: Invalid file control request according to the CICS file control table. A REWRITE, or a DELETE, was issued but previous READ with UPDATE command was issued. A READNEXT, READPREV, ENDBR, or RESETBR was issued but no previous STARTBR was issued. A DELETE command is issued for a non VSAM file. A DELETE command with the RIDFLD option specified is issued for a VSAM data set when a READ UPDATE command is outstanding. Following a READ UPDATE command for a data set, a WRITE or READ UPDATE command is issued for the same data set before exclusive control is released. The data area specified in the RIDFLD is not the same one in all the commands of a browse. An attempt is made to start a browse with a REQID already in use for another browse. PROGRAM CONTROL: A RETURN command with the Commarea option is issued in a program that is not at the highest logical level. A RETURN command with the Transid option is issued in a task that is not associated with a terminal.
AEIQ	IOERR An I/O error occurred; contact the systems team.

AEIR	NOSPACE	Occurs if no space is available for adds.
AEIT	ENDFILE	Occurs if an end-of-file condition is detected during a browse.
AEIU	ILLOGIC	Occurs if a VSAM error occurs that does not fall within one of the other CICS/VS response categories.
AEIV	LENGERR	<p>FILE CONTROL:</p> <p>The length option is not specified for an input (without the set option) or output operation involving variable-length records.</p> <p>The length specified for an output operation exceeds the maximum record size; the record is truncated.</p> <p>The length of a record read during an input operation (with the into option) exceeds the value specified in the length option; the record is truncated, and the data area supplied in the length option is set to the actual length of the record.</p> <p>An incorrect length is specified for an input or output operation involving fixed-length records.</p> <p>TEMPORARY STORAGE CONTROL:</p> <p>Occurs if the length of the stored data is greater than the value specified by the length option for move-mode input operations.</p> <p>TERMINAL CONTROL:</p> <p>Occurs if the length of data received by an input request that specifies the into option exceeds the value specified by the length or to length option.</p>
AEIW	QZERO	Occurs when the destination (QUEUE) accessed by a READQ TD is empty.
AEIZ	ITEMERR	Occurs when the item number specified or implied by a READQ TS, or a WRITEQ TS with the rewrite option, is invalid; or if the QUEUE referred to in a WRITEQ TS cannot be found.
AEIO	PGMIDERR	Occurs if a program or map cannot be found in the CICS processing program table (PPT) or is disabled.
AEI8	TSIOERR	Occurs if there is an unrecoverable temp storage I/O error.
AEI9	MAPFAIL	Occurs if the data to be mapped has a length of zero or does not contain a set buffer address (SBA) sequence.
AEXL	DISABLED	Occurs when a selected data set is not available.
AEYG	JIDERR	Occurs if the specified journal file identifier does not exist in the journal control table (JCT).

AEYH	QIDERR	Occurs when the symbolic name identifying the QUEUE to be used with TS or TD requests cannot be found.
AEY9		EXEC command not supported by current system. Function not available, usually in DB2 programs.
AICA		A runaway task condition has been detected. This indicates a logical loop.
APCT		A program cannot be located in the PPT.
ASRA		A program interrupt has occurred. Either non-numeric data was found in a numeric field or else addressability has not been properly established.
ATDD		The destination that was specified in a transient data request is disabled. This could be due to the data set being full.