Unit 1. Programming Components

Overview

- **CICS Introduction**
- **CICS Command Syntax**
- **Basic CICS commands**
- **Application Program Preparation**

Figure: 1-1. Overview

Notes :

CICS

- **CICS is a database/data communication system(DB/DC).**
- **CICS can be termed as an Online Transaction Processing System(OLTP).**
- **CICS forms a layer in between the application programs and the operating system and act as an interface between the two.**

Figure 1.2. CICS

Notes :

CICS = Customer Information Control System

CICS began as one of the original Online Transaction Processing (OLTP) manager.

A typical Transaction –

Receives a message/data from a terminal.

Accesses data from a File or Database.

Sends a reply back to Terminal.

CICS Today: Client / Server

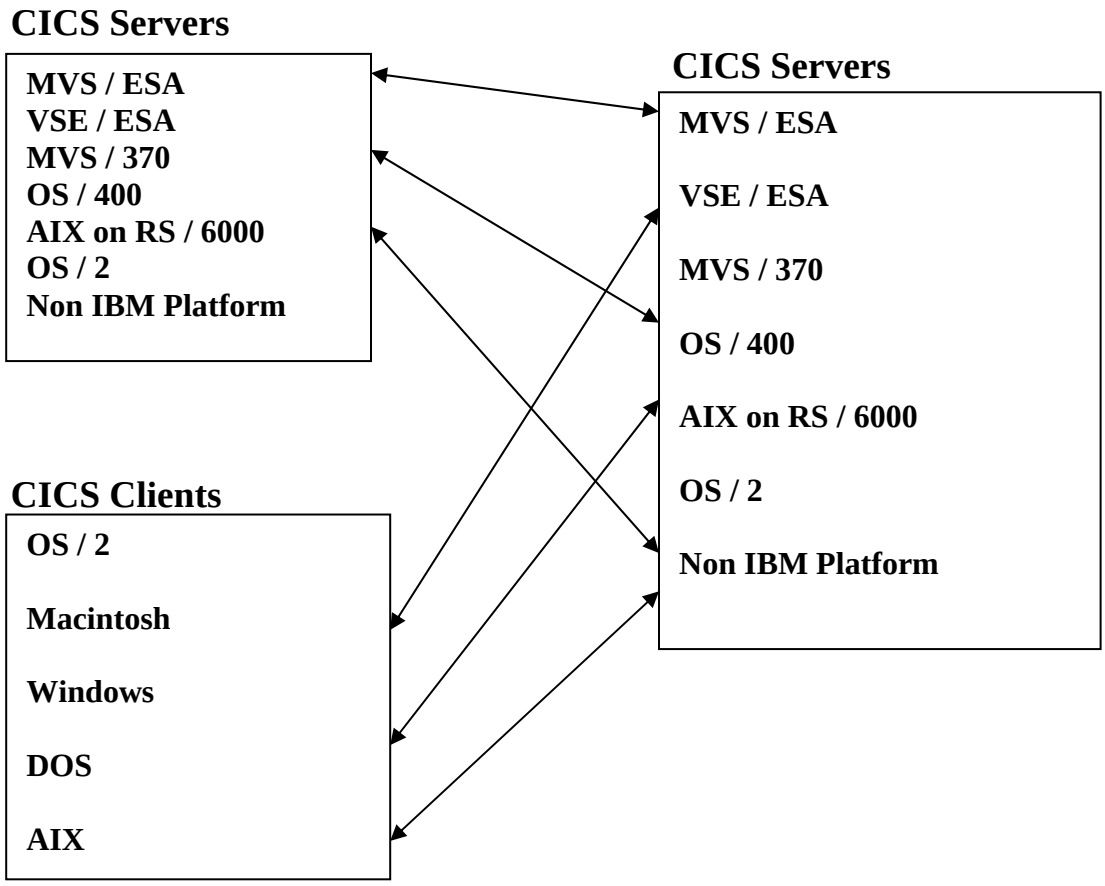

Figure: 1-3. CICS Today : Client / Server

Notes :

CICS supports all major IBM hardware product lines and several non-IBM hardware IBM products.

CICS Command Syntax

- **CICS Command format**
- **Argument values**
- **Data types**

Figure: 1-4 CICS Command Syntax

Notes :

CICS Command Format

```
EXEC CICS function
    [option (argument value)]
    [option (argument value)]
    ……
    …….
END-EXEC.
```

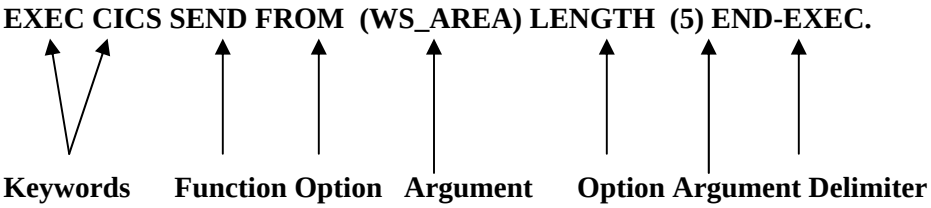

Figure: 1-5 CICS Command Format

Notes :

A CICS command consists of a keyword phrase, delimiter, function, options and their argument values.

The EXEC CICS commands must always start in column 12 or greater for COBOL CICS programs.

PL/I uses a semi-colon as a delimiter instead of END-EXEC.

Argument Values

Data Value	**The data value is an actual value or name of a field which contains that value. It could be a literal constant value or the field name in which the value is defined.**
Data Area	**It is a data name or field name which is defined in the working storage section. It is used for specifying names such as input/output areas or key fields.**
Name	**The name is a literal constant or the name of a data area defined in the working storage section. It is used for defining dataset names , mapnames etc.**
HHMMSS	**It indicates a decimal constant or the data name of S9(7) comp-3 field. It is used for time the expression.**
Pointer-ref	**It is the name of the base locator for linkage (BLL) cell, which is defined as a full word binary field(PIC S9(8) comp).It is used to establish addressability.**
Label	**A label is a Cobol paragraph or section name and is used for passing control to the paragraph or section in the procedure division.**

Figure: 1-6. Argument values

Notes :

Data Types

CICS COBOL	SQL COLUMN
PIC S9(4) COMP	SMALLINT
PIC S9(8) COMP	INTEGER
PIC S9(5) COMP-3	DECIMAL
PIC X(5)	CHAR
	VARCHAR

Figure: 1-7. Data Types

Notes :

Basic CICS Commands

- **Terminal Control Commands**

  SEND

  RECIEVE

- **Positioning the Cursor**

  SEND CONTROL

- **Data and Time Formatting Commands**

  ASKTIME

  FORMATTIME

- **Program Termination**

  RETURN

Figure:1-8. Basic CICS Commands

Notes :

Terminal Control Commands(SEND)

SEND Command

Syntax :

```
EXEC CICS SEND
        FROM(data_area)
        [LENGTH (data_value)]
        [DEST(name)]
END-EXEC.
```

Figure: 1-9. Terminal Control Commands(SEND).

Notes :

Example:

```
 PROCEDURE DIVISION.
*
 MOVE   'INPUT DATA IS TOO LONG'   TO    MSG.
 EXEC CICS  SEND
                    FROM(MSG)
                    LENGTH(22)
 END-EXEC.
 ……
 ……
```

Terminal Control Commands(RECEIVE)

RECEIVE Command:

Syntax :

```
EXEC CICS RECEIVE
        [INTO (data area) |
        [LENGTH (data area))]
  END-EXEC
```

Figure: 1-10. Terminal Control Commands(RECEIVE)

Notes :

Example:

```
PROCEDURE DIVISION.
*
MOVE 9 TO INL.
EXEC CICS  RECIEVE
                    INTO(WS-AREA)
                    LENGTH(INL)
END-EXEC.
……
……
```

Positioning the Cursor

SEND CONTROL Command:

Syntax :

```
EXEC CICS SEND CONTROL
    CURSOR(data_value)
     [ERASE | ERASEAUP]
     [PRINT]
     [FREEKB]
     [ALARM]
     [FRSET]
  END-EXEC
```

Figure: 1-11. Positioning the Cursor.

Notes :

Example:

```
 PROCEDURE DIVISION.
*
 MOVE   'INPUT DATA IS TOO LONG'   TO     MSG.
 EXEC CICS  SEND  CONTROL(1760)
 END-EXEC.
 EXEC CICS  SEND  FROM(MSG)
                  LENGTH(22)
 END-EXEC.
 ……
 ……
```

Data and Time Formatting Commands(ASKTIME)

ASKTIME Command:

Syntax:

```
EXEC CICS ASKTIME
     [ABSTIME(data_area)]
END-EXEC.
```

Figure: 1-12. Date and Time Formatting Commands(ASKTIME)

Notes :

Example:

```
 PROCEDURE DIVISION.
*
  EXEC CICS  ASKTIME  ABSTIME(TSTAMP)
                      DATESEP
                      MMDDYY(OUTDATE)
 END-EXEC.
 ……
 ……
```

Data and Time Formatting Commands(FORMATTIME)

FORMATTIME Command:

Syntax:

```
EXEC CICS FORMATTIME
     [ABSTIME(data_area)]
     [YYMMDD(data_area)]
     [YYDDD(data_area)]
     [YYDDMM(data_area)]
     [DDMMYY(data_area)]
     [DATE(data_area)]
     [DATEFORM(data_area)]
     [DATESEP(data_value)]
     [DAYCOUNT(data_area)]
     [DAYOFWEEK(data_area)]
     [DAYOFMONTH(data_area)]
     [MONTHOFYEAR(data_area)]
     [YEAR(data_area)]
     [TIME(data_area)]
     [TIMESEP(data_value)]
END-EXEC.
```

Figure: 1-13. Date and Time Formatting Commands(FORMATTIME)

.

Notes :

Example:

```
 PROCEDURE DIVISION.
*
  EXEC CICS  FORMATTIME  ABSTIME(WS-AREA)
                         DATESEP('-')
                         MMDDYY(OUTDATE)
 END-EXEC.
 ……
```

……

Program Termination

RETURN Command:

Syntax:

```
EXEC CICS
        RETURN
END-EXEC.
```

Figure: 1-14. Program Termination.

Notes :

Example:

```
 PROCEDURE DIVISION.
*
  EXEC CICS  RETURN
 END-EXEC.
 ……
 ……
```

The above example returns control to CICS region.

Application Program Preparation

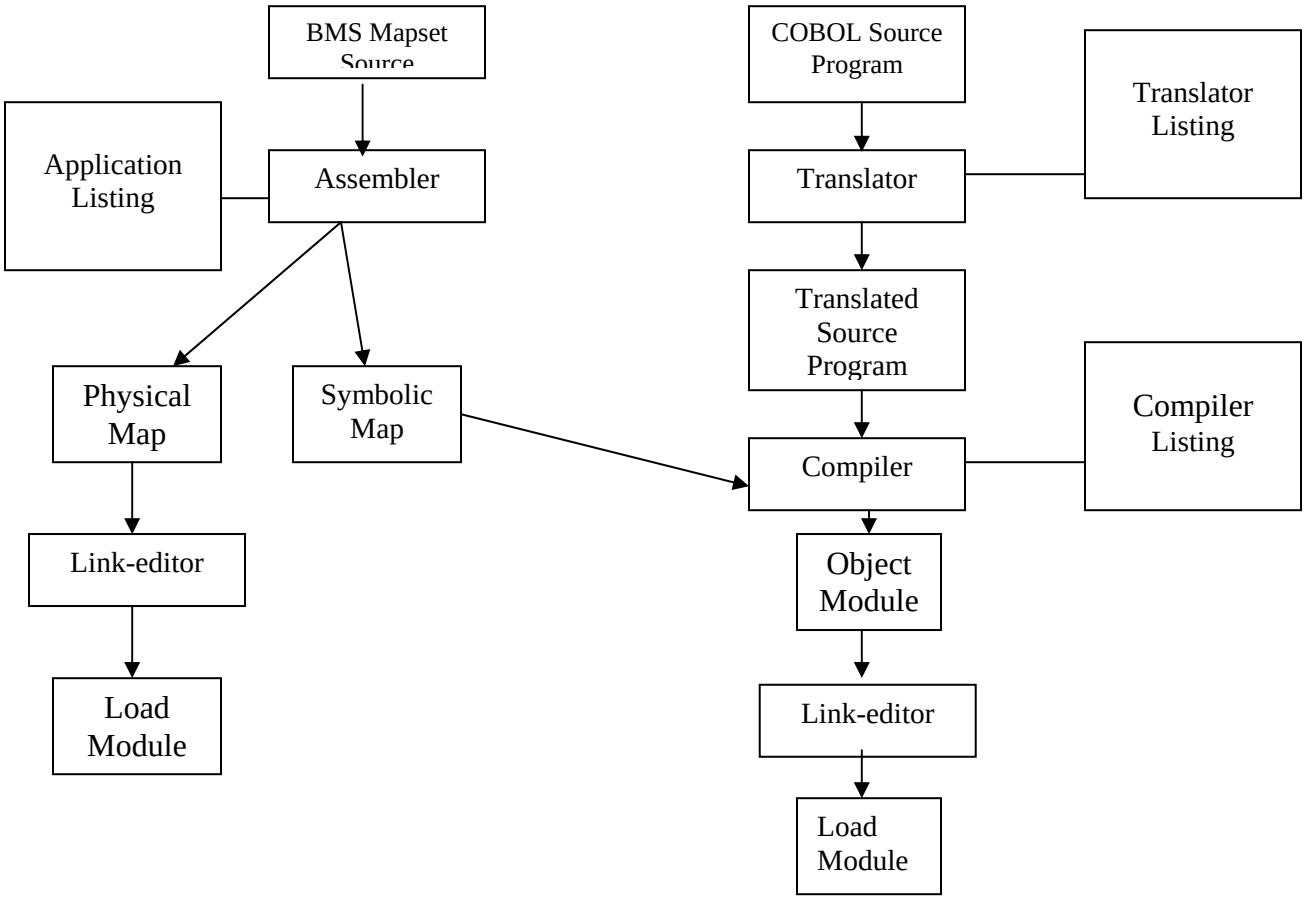

Figure: 1-15. Application Program Preparation.

Notes :

The TRANSLATOR recognizes EXEC CICS statements. They are commented out and replaced with statements in the appropriate language. In our case, COBOL MOVE instructions and a CALL are inserted and passed on to the COBOL compiler.

The DB2 Precompiler recognizes EXEC SQL statements which it comments out and replaces with COBOL PERFORM and CALL statements (in our case).

Testing

CEMT SET PROGRAM(PROG1) PHASEIN

Or

CEMT S PR(PROG1) PH

CEMT SET PROGRAM(PROG1) NEWCOPY

Or

CEMT S PR(PROG1) NE

Figure: 1-16. Testing

Notes :

After making changes to a program a new version replaces the old version. But CICS which is currently executing has no way of knowing automatically. The CICS processing program table still points to the old version.

To avoid testing with the old version you must use the CICS provided CEMT transaction to update the pointer to the program.

Debugging with CEDF

CICS provided debugging transaction, CEDF (Command Execution Diagnostic Facility) trace search command in the application.

To invoke CEDF :

- **First type CEDF in the upper left corner of a cleared screen.**
- **Then clear the screen and type in the transaction ID you want to test.**

Figure: 1-17. Debugging with CEDF

Notes :

CEDF allows the user to debug an Application Program. But it debugs the CICS commands and also gives the status of Debugged Command.

Conversational vs. Pseudoconversational

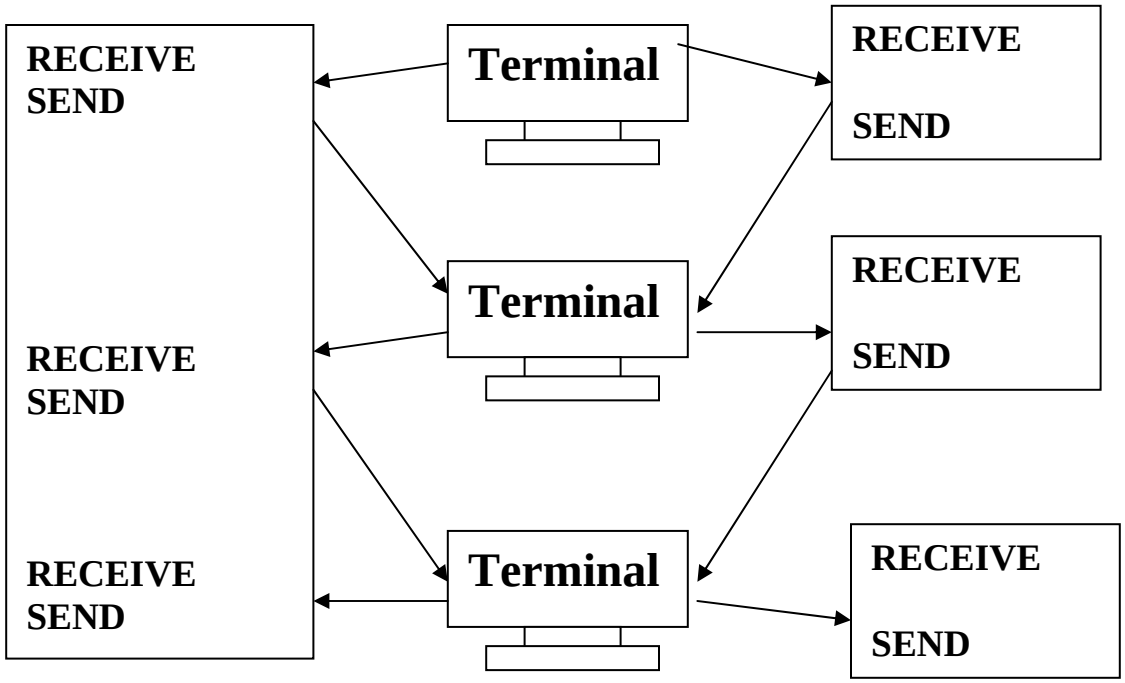

Figure: 1-18. Conversational vs. Pseudoconversational.

Notes :

A conversational program is a program, which begins execution when a transaction id invokes it and terminates when it encounters a RETURN statement, whereby the control is returned back to CICS.

A pseudoconversational program is one which is divided into multiple tasks, whereby each task Issues a RETURN command along with a Transaction id showing its intention to get back in the transaction. For pseudoconversational transactions storage and other resources are not held up during operator think time.

COBOL Sections for CICS

IDENTIFICATION DIVISION.
PROGRAM_ID.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.

DATA DIVISION.
FILE SECTION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.

PROCEDURE DIVISION.

Figure: 1-19. COBOL Section for CICS.

Notes :

Terminate CICS program logic with the EXEC CICS RETURN Command.

Include a COBOL GOBACK to avoid a compiler warning or error.