Unit 3. MAPS

Objectives

- **Objectives of BMS**
- **BMS Maps**
- **Map Fields**
- **Modified Data Tag**

Figure: 3-1. Objectives

Notes:

Objectives Of BMS (Basic Mapping Support)

To remove the device independent codes from an application program and places them in maps.

To remove constant information like title headers etc from an application program and placing them in maps.

Constructing NMDS required for producing the desired screen.

To provide access to data fields of NMDS using symbolic names, Which allows the repositioning of fields without modifying the Applications programs.

Provide text-handling capability.

To provide the terminal paging facility, which allows combination of several small mapped areas into one or more pages of output.

To provide the message routing facility, which allows sending of messages to one or more terminals.

Figure: 3-2. Objectives of BMS

Notes:

BMS is a standard facility of CICS, which deals with formatted screen operations.

CICS programmers have a good deal of control over the way information is presented on 3270 Type devices. For example, you can determine:

How data is arranged on the screen.
How features, such as colour, are used to highlight and differentiate screen fields and improve Readability.

With Basic Mapping Support, these 3270 data streams are easy to deal with.

BMS maps

- **Map is a screen defined through BMS.**
- **BMS map is nothing but a program written in Assembly language.**
- **A set of Assembler macros(BMS macros) are provided by CICS for the BMS map coding.**
- **A map represents a BMS coding for a screen panel.**
- **A mapset contain different maps.**

Figure: 3-3. BMS Maps

Notes:

Normally a screen is divided into several fields by the program sending a screen map.

It is possible to have a screen with no fields i.e. with no attribute characters. The operator then types a character string, typically a transaction code and account number, into the blank screen and presses enter. BMS is not used to interpret this type of input.

A mapset is collection of BMS maps link edited together.

Types of Maps

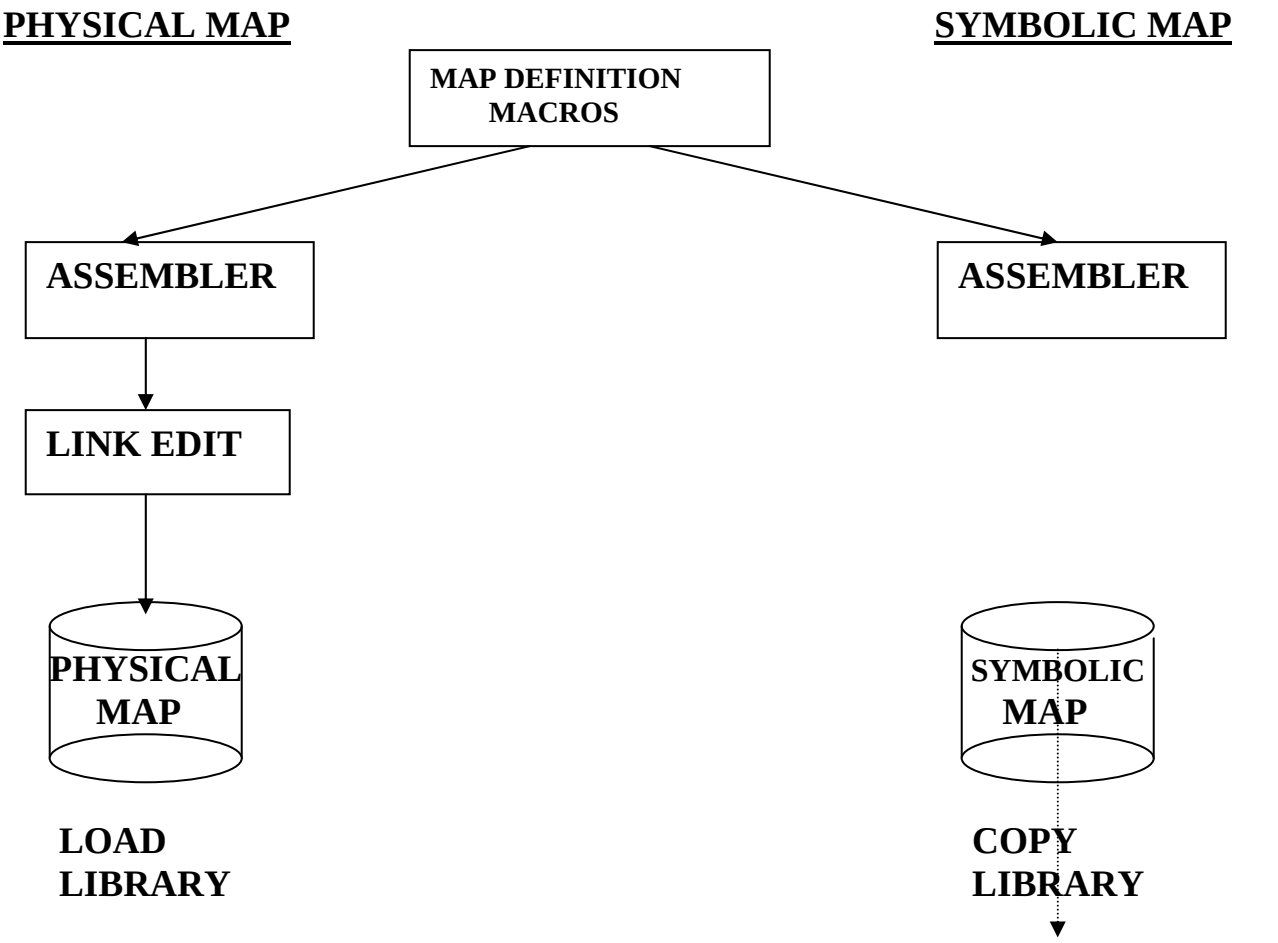

Figure: 3-4. Types of Maps

Notes:
Physical Map and Symbolic Map

Map Contents
The physical map is a table of screen display field information.
The symbolic map is a structure of field definitions in the specified programming language.
Where Stored
The physical map resides in the program load library.
The symbolic description map resides in the COPY library.
Where used
The physical mapset is loaded when an executing program specifies the name of the mapset in a SEND or RECEIVE command.
The symbolic map is placed in the source program by the compiler.

Physical Map

Physical Map defines

- **Starting Position**
- **Length**
- **Field Characteristics (Attribute byte)**
- **Default data for each field**

It Allows BMS to add BCC's and commands for output in order to construct an o/p NMDS. A Physical map will be in the form of a load module.

Figure: 3-5. Physical Map

Notes:
The physical map describes the format of the screen that the enduser uses to input data.

The physical map represents Load module.

The physical map contains the screen position, length, attribute and color for each field that is Displayed at the terminal, along with the terminal type.

Symbolic Map

A symbolic map is a copy library member which can be included in the application program for defining screen fields.

It defines the map fields used to store variable data referenced in COBOL program.

For each field name in DFHMDF macro BMS creates 3 fields for i/p and 3 fields for o/p by placing a character suffix to the original field name.

Figure: 3-6. Symbolic Map

Notes:

The symbolic map starts with the 01 level definition of map name defined in the DFHMDF macro with suffix ‘I’ for input map an ‘O’ for o/p map.

Next a filler of pic x(12) which is the TIOAPrefix and is created if the option TIOAPFX=YES is specified in DFHMSD macro.

The symbolic map represents copy book for map fields.

A program using a map to send and receive data from a terminal, will copy a symbolic map into its working storage.

Symbolic Map Fields

Name + L → Half word binary field pic s9(4) comp.

Name + F → Flag byte, for i/p field X'80' if modified X'00'.

Name + A → Attribute byte for both i/p and o/p fields.

Name + I → I/p data field X'00' will be placed if no data is entered.

Name + O → The o/p data field.

Figure: 3-7. Symbolic Map Fields

Notes:
For each field, there are 3 definitions for input maps:

NameL This is used to store the number of characters the user typed into the field
NameF This is a flag byte and is normally set to X'00'.
NameI This contains the input data as read from the display.

And 2 definitions for output maps:

NameA This gives programmers access to the attribute byte so they can change the display characteristics of a field.
NameO This stores the output data to be sent to the display.

Map Definition Macros

There are 3 CICS supplied macros for coding BMS maps.

DFHMSD Mapset definition macro.

DFHMDI Map definition macro.

DFHMDF Field definition macro.

Figure: 3-8. Map Definition Macros

Notes:
You can define a CICS map using BMS Assembler macros, the three macros shown above.

Together, they are used to define the overall characteristics of the map and the way in which data is presented. This includes:
The name of the map, its size and the name of the mapset to which it belongs.
The language of the program it will be used in Screen and keyboard control information.

DFHMSD Macro

Syntax:

```
Mapsetname   DFHMDF   TYPE=&SYSPARM,                    x
                      MODE=INOUT,                       x
                      LANG=COBOL,                       x
                      STORAGE=AUTO,                     x
                      TIOAPFX=YES,                      x
                       CNTL=(FREEKB,FRSET,PRINT)
```

Figure: 3-9. DFHMSD Macro

Notes:

Mapsetname is of 1 to 7 characters in length.

DFHMSD macro is used to define a mapset and its characteristics or to end a mapset definition.

DFHMDI Macro

Syntax :

```
Mapname   DFHMDI   SIZE=(line,column),          x
                   LINE=number,                 x
                   COLUMN=number,               x
                   JUSTIFY=LEFT
```

Figure: 3-10. DFHMDI Macro

Notes:

Mapname is of 1 to 7 characters in length.

DFHMDI macro is used to define a map and its characteristics in a mapset.It can be used often as you wish within one DFHMSD macro.

DFHMDF Macro

Syntax:

```
Fieldname   DFHMDF   POS=(line,column),                    x
                     LENGTH=number,                        x
                     JUSTIFY=RIGHT,                        x
                     INITIAL=Char data,                    x
                     ATTRB=(UNPROT|PROT,NUM,FSET,IC)
```

Figure: 3-11. DFHMDF Macro

Notes:
DFHMDF is used to define the fields with attributes.

Modified Data Tag

It is a 1-bit attribute character.

If

MDT is off(0) **indicates that the field has not been modified at the terminal.**

MDT is on(1) **Indicates that the field has been modified at the terminal only it is on, the data of the field will be sent by the terminal to the host program.**

Figure: 3-12. Modified Data Tag

Notes:
MDT is a 1 bit attribute character of a BMS field. When it is set on, CICS will transmit the data contained in the associated map field.

Setting and Resetting MDT

When a user modifies a field MDT will automatically be set to ON(1)

If CNTL=FRSET is specified in the DFHMSD macro or DFHMDI macro, when the map or mapset is sent to the terminal MDT will be reset to 0 or OFF for all fields of the map or mapset.

If FSET is specified in the ATTRB option of the DFHMDF macro, when the map is sent to the terminal, MDT will be set to 1 or ON for the fields regardless of whether the field has been modified by the user.

Figure: 3-13. Setting and Resetting MDT

Notes:
When the application program moves the DFHBMFSE to the attribute character of the field then also MDT in ON state**.**