

Basic Mapping Support

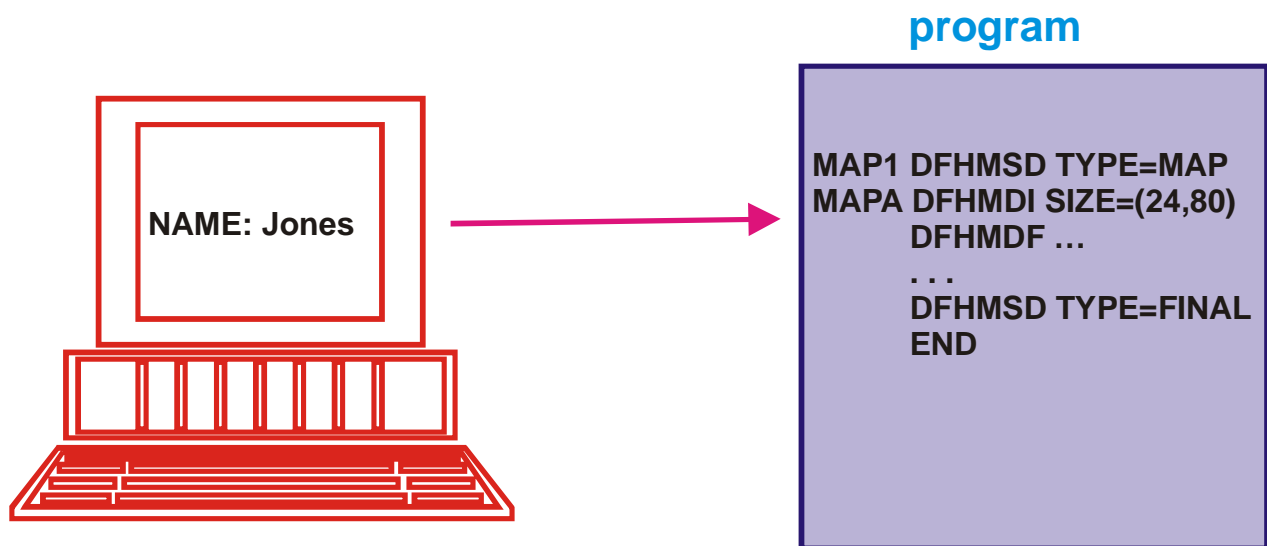
Introduction	48
3270 Concepts	51
BMS Modules	53
DFHMSD	53
DFHMDI	56
DFHMSD	57
The Symbolic MAP	60
DFHBMSCA copybook	62

Introduction

Basic Mapping Support is a (primitive) CICS facility which enables the programmer to define the way in which information is presented to the user on a 3270 screen.

Producing screen definitions is a two-stage process. First you have to code the locations of the fields using a series of ancient Assembler language macros. These define the MAP layout; a map is the name given to a CICS screen definition.

(Alternatively, you could use a map generator such as SDF II or BMS/GT to do the hard work.)

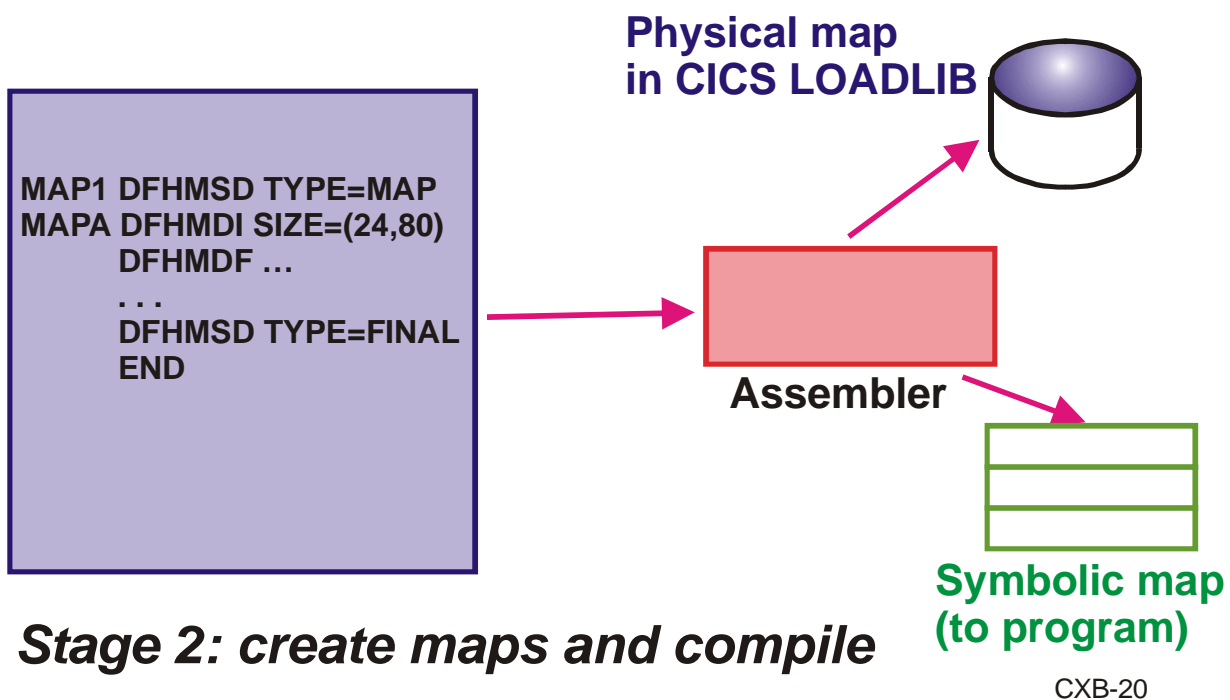


Stage 1: design and code map

CXB-10

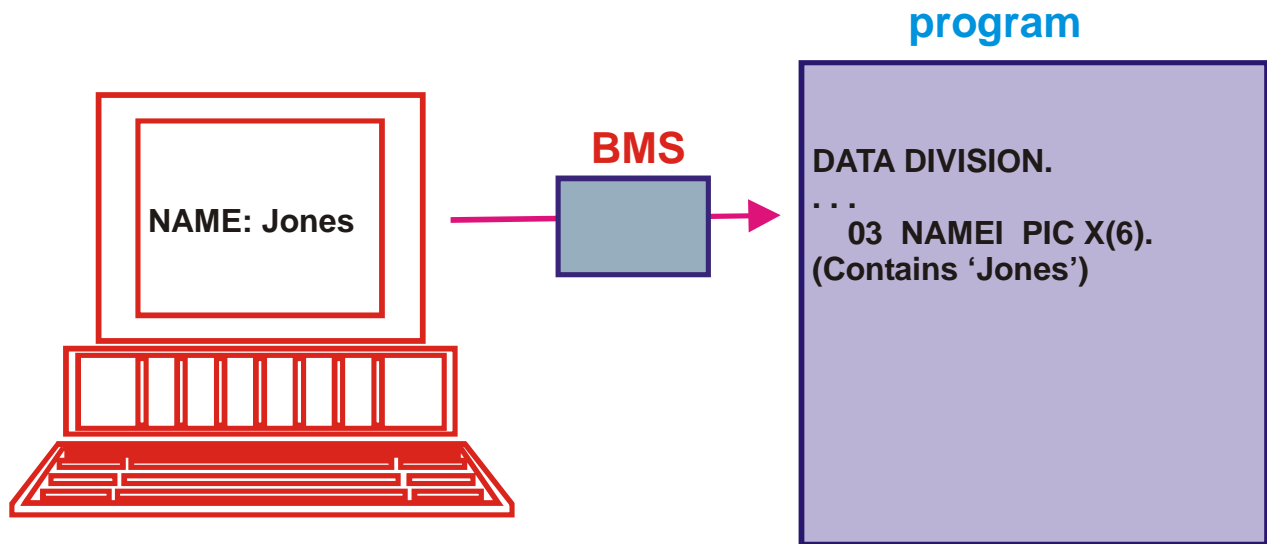
The second stage of the process is to take the BMS macro definitions and then Assemble and Link edit them. This produces two forms of output:

- a *physical* map or object program which is linked to the CICS core image or load library and has a PPT entry similar to that of an application program.
- a *symbolic* map or source deck in COBOL, ASSEMBLER, PL/1, or RPGII defining the names of the fields specified in the definition. These must be incorporated in the application program handling the map prior to compilation.



Menu Functions Utilities Help									
BROWSE OAK001.LOADLIB							Browse substituted		
Command ==>							Scroll ==> CSR		
Name	Size	TTR	Alias-of	AC	AM	RM	---- Attributes ----		
. *****	000018C0	005C06		00	31	ANY			
. CCINQM	00000448	00042B		00	24	24			
. CCMENM	000002C8	000432		00	24	24			
. CCUPDM	00000498	000439		00	24	24			
. DFHPRGA	000015B0	003917		00	31	ANY			
. PRGCMMSG	00001630	000506		00	31	ANY			
. PRGCTIM	00001310	000514		00	31	ANY			

Once the physical and symbolic maps have been produced, you can compile and link edit your application program. This brings in the BMS definitions which remain behind-the-scenes and are transparent to the end-user.



BMS mapping

CXB-30

Code within the application program moves appropriate values into the symbolic fields and issues the CICS command to *write* the screen. The two main benefits provided by BMS are:

- *device independence* No intimate knowledge of the hardware characteristics of the particular model of the terminal is required by the programmer.
- *format independence* A rearrangement of the data fields on the screen does not alter application program logic. It might not even alter the symbolic map.

3270 concepts

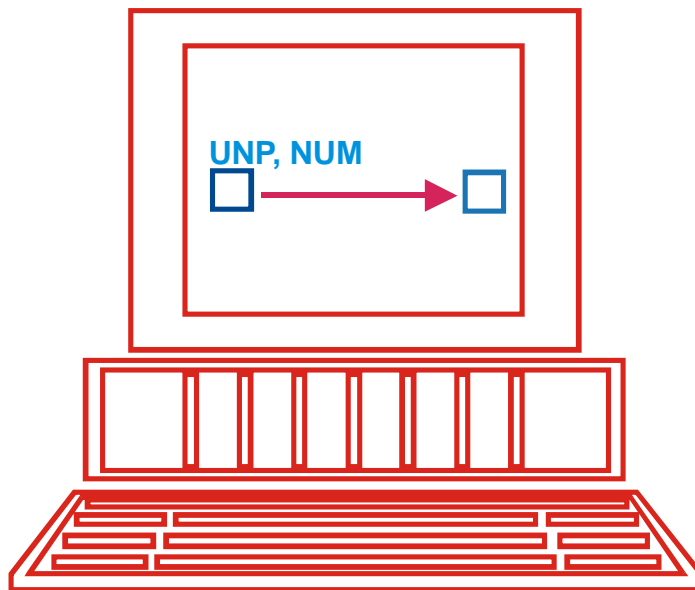
The standard 3270 terminal has 24 lines of 80 characters available for definition in a BMS map.

The definition will divide the 1920 character positions into *fields* which are either:

- *available* for keying
- or
- *unavailable*

Each *field* has its characteristics specified by an *attribute* byte which precedes the field data and is located at a specified character position on the screen. The attribute byte itself is non-visible *and* unavailable for overkeying.

The characteristics defined by the attribute byte apply to the data following (left to right), not just for the length specified, but *until another attribute byte is specified*. This extra attribute byte is known as a stopper byte, and frequently has the Autoskip property – move the cursor to the next input field.



CXB-40

Scope of attribute bytes

Extended attributes are used for color and highlighting, but are not physically present on a screen.

Some of the characteristics which may (in combination) be associated with a field are:

UNPROTECTED data may be keyed

PROTECTED data may not be keyed

BRIGHT intensity of display

NORMAL

DARK DARK means invisible (good for passwords!)

FSET *Modified Data Tag on* - the field will behave as though data has been keyed into it and will therefore be re-transmitted as input

NUMERIC numeric data only may be keyed. This is not as useful as it first appears, since it still permits the entry of certain non-numeric special characters.

ASKIP causes the cursor to tab to the next *UNPROTECTED* field on encountering this field (equivalent to *NUMERIC*, *PROTECTED*)

IC positions the cursor at this field initially

Some of the above are mutually exclusive.

Several other attributes are available for terminals with selector pens, color, and other features; the above encompass the bulk of general usage.

When an operator interacts with an application program (by means of an *ATTENTION* or *FUNCTION KEY* - e.g. ENTER, CLEAR, PAn, PFn), the program will receive the transmitted information into its *symbolic definition*. It will then respond accordingly after checking and altering the data and/or attributes or by transferring control to another program.

Any map characteristics not altered by the program will default to the original assembled map definition.

BMS macros

The BMS Assembler macros must be coded to define a map. Essentially, three are used:

- DFHMSD define a *mapset*
- DFHMDI define a *map*
- DFHMDF define a *field*

DFHMSD

This macro instruction defines a *mapset* or collection of related maps which may be stored together. Alternatively, you may define each *mapset* containing a single map.

The macro appears in a definition in two places - at the beginning and at the end. Typically, you might have:

```
MAPS1    DFHMSD TYPE=MAP,MODE=INOUT,CTRL=(FREEKB),      X
          LANG=COBOL,TIOAPFX=YES
          .
          .
          DFHMSD TYPE=FINAL
          END
```

Notice that the *Assembler* language coding rules apply for column positions:

- *Label* starts in column 1
- *Operation* starts in columns 2 to 16 (typically 10)
- *Operands*
 - follow the operation and are separated by one or more blanks
 - must be concatenated by a comma (,) with no intervening blanks
 - there is no comma following the last blank
- *Comments* follow the operands and are separated by one or more blanks
- *Continuation*: non-blank in column 72, next operands start on column 16 for the subsequent line

Many macro parameters are available. The most commonly used are:

TYPE=	DSECT	generate only the symbolic map
	MAP	generate only the physical map
	FINAL	must be present to end a mapset
MODE=	IN	generate an input map
	OUT	generate an output map (default)
	INOUT	most common - both input and output symbolic maps are generated with the output definition redefining the input
CTRL=	FREEKB	unlocks the keyboard after the map is displayed
	FRSET	reset any modified data tags - this ensures that any operator-entered fields will not be re-transmitted!
ALARM		the audible alarm feature
LANG=	COBOL	the programming language used
	ASM	for this application
	PLI	NB - the default is ASSEMBLER
	RPG	
TIOAPFX=	YES	a <i>filler</i> is included at the beginning of the symbolic definition. Mandatory for command level applications.

The above *skeleton* is sufficient for most purposes, but other parameters provide for color, extended attributes, data stream formatting, and other relatively rare features.

Maps produced by CICS Transaction Server for OS/390 Release 3 (and later versions) include an **ADS descriptor** in the output load module. This enables the BMS Application Data Structure to be interpreted without requiring your program to include (COPY) the relevant copybook at compile time. This structure is used by your application program for the data in SEND and RECEIVE MAP requests.

The ADS descriptor contains:

- a header with general information about the map
- a field descriptor for every field that appears in the ADS

The ADS descriptor is generated for all maps. If you use the DSECT=ADSL option to select the long form of the ADS, all the fields are aligned on 4-byte boundaries. The long form of the ADS is needed by the 3270 Bridge when using an interface to MQSeries.

DFHMDI

This macro is used to define a single map. It follows the first DFHMSD and can have a wide variety of parameters. Many of these are identical to those of the DFHMSD macro. Typically, it appears as:

```
PPMSM01    DFHMDI    SIZE=(24,80)
```

where PPMSM01 is the map name. Other parameter options include:

COLUMN=xx specifies the column at which the map is placed. This provides a left or right hand margin.

```
+-----+
| >>--map--DFHMDI-----> |
|                               |
| <-,-----+ <-,-----+ |
| >-----+-----+-----+ < |
| | -SIZE=(line,column)----- | | -PS=--+BASE-+----- | | | |
| | -TERM=type----- | | +-psid+ |
| | <-,-----+ | | <-,-----+ |
| | -CTRL=(--+-----)--- | | -VALIDN=--+-----+ |
| | | -PRINT-- | | | -MUSTFILL-- |
| | | -length- | | | -MUSTENTER- |
| | | -FREEKB- | | | -TRIGGER--- |
| | | -ALARM-- | | | +-USEREXIT--+ |
| | +-FRSET--+ | | | -COLUMN=number----- |
| | +-NO-----+ | | | -LINE=number----- |
| | -EXTATT=+-----+ | | | -FIELDS=NO----- |
| | | -MAPONLY- | | | <-,-----+ |
| | +-YES-----+ | | | --MAPATTS=(attr)+----- |
| | +-DEFAULT--+ | | | <-,-----+ |
| | -COLOR=+-----+ | | | --DSATTS=(attr)+----- |
| | +-color---+ | | | -OUTLINE=--+BOX-----+ |
| | +-NO--+ | | | | <-,-----+ |
| | -CURSLOC=+-----+ | | | +-(-+-----)-+ |
| | +-YES+ | | | | -LEFT-- |
| | +-OFF-----+ | | | | -RIGHT- |
| | +-HIGHLIGHT=+-----+ | | | | -OVER-- |
| | | -BLINK----- | | | | +-UNDER--+ |
| | | -REVERSE--- | | | +-NO--+ |
| | | -UNDERLINE--+ | | | -SOSI=+-----+ |
| | | | | | | +-YES+ |
| | | | | | | +-YES--+ |
| | | -TRANSP=+-----+ | | | -TRANSP=+-----+ |
| | | +-NO--+ | | | +-NO--+ |
| | | -JUSTIFY=BOTTOM-----+ |
+-----+
```

DFHMDF

In normal usage, it is this field definition macro which contains the greatest number of parameters. Except in the most sophisticated of applications, both DFHMDS and DFHMDSI remain virtually static, as we have seen.

An example of DFHMDF is:

```
EMPNO    DFHMDF POS=(14,9),LENGTH=8,                X
          INITIAL='00387344',ATTRB=(UNPROT,NUM,FSET)
```

where EMPNO is, for example, the *symbolic name* of the field. If no name is entered, no reference is generated in the symbolic map. This is frequently used for title fields containing fixed literals.

The most useful parameters are:

POS=(14,9) defines the start position of the *attribute byte* preceding the data field. The position may appear as a number (e.g. 93) [absolute position from 0 (top left)].

The start positions must appear in ascending sequence, may not overlap, and can also be absolute numbers or row-and-column ordered pairs. Use absolute numbers only at your own risk.

LENGTH=8 defines the length of the data portion of the field (excluding attribute). This only controls the number of bytes which can be mapped onto the symbolic map. Remember that the field is delimited by the next attribute on the screen. For this reason, it is common to define a 1 byte stopper field immediately following a data entry field.

INITIAL='00387344' specifies the initial value of a field.

Do not use XINIT to define hexadecimal data unless you know a great deal about the 3270 control characters. In any case, avoid all values less than X'40'.

FSET treats the field as though data had been entered by the user; the field will be retransmitted on input.

```

+-----+
|>>+-----DFHMDf----->|
|+-fld-+|
|
|<-,------+|
|>----->|
|
| -POS=-+number-----+
|   +-(line,column)-+
| -LENGTH=number-----+
| -JUSTIFY=(--+-----+)|
|           | -LEFT-- | +-,-+-BLANK-+---+
|           +--RIGHT-+   +-ZERO---+
|
| +-INITIAL='char-data'-----+
| | -XINIT=hex-data----- |
| |                       (1) |
| +-GINIT='DBCS-characters'-----+
|   +-ASKIP-----+
|
| -ATTRB=(--+-----+)|
|           | -PROT----- | | <-----+ |
|           +--UNPROT-+---+ +-,-+-BRT-+---+---+
|                   +- ,NUM-+ | -NORM- | | -,DET-- |
|                               +-DRK---+ | -,IC--- |
|                                       +- ,FSET-+
|
|           +-DEFAULT-+
|
| -COLOR=-+-----+
|         +-color---+
|         +-BASE-+
|
| -PS=-+-----+
|       +-psid-+
|         +-OFF-----+
|
| -HIGHLIGHT=-+-----+
|           | -BLINK----- |
|           | -REVERSE--- |
|           +-UNDERLINE-+
|           <-,------+
|
| -VALIDN=(--+-----+)|
|           | -MUSTFILL-- |
|           | -MUSTENTER- |
|           | -TRIGGER--- |
|           +-USEREXIT--+
|
| -GRPNAME=group-name-----+
| -OCCURS=number-----+
| -PICIN='value'-----+
| -PICOUT='value'-----+
| -OUTLINE=-+BOX-----+
|           | <-,------+ |
|           +-(-+-----+)-+
|           | -LEFT-- |
|           | -RIGHT- |
|           | -OVER-- |
|           +-UNDER-+
|
|         +-NO---+
|
| -SOSI=-+-----+
|       +-YES-+
|       +-YES-+
|
| -TRANSP=-+-----+
|       +-NO---+
|
| +-CASE=MIXED-----+
|
| (1) DBCS characters start with a shift-out character X'0E' and end
|     with a shift-in character X'0F'.
|
+-----+

```

ATTRB=(UNPROT,NUM) specifies the combination of initial attributes of the field as described in 3270 concepts. Some are mutually exclusive and will cause assembly errors if they appear together.

JUSTIFY=

RIGHT	specifies how screen data is mapped to the
LEFT	symbolic definition and which filler characters are used.
BLANK	This defaults to either RIGHT,ZERO for numeric fields
ZERO	or LEFT,BLANK for alphanumeric fields

OCCURS=xx an array of entries is to be generated for the field.
Use a subscript to access an individual entry.

GRPNAME=ADDRS the field appears as part of a group field **ADDRS**.
This must be specified on each element of the group. Date is only and example. It concatenates fields with no intervening attributes.

OCCURS and **GRPNAME** are *mutually exclusive*, which limits their value. Frequently we would like to present columns of group fields on screen.

PICIN='9V999' permits the use of edit facilities
PICOUT='ZZB.99' for the field in either the input or output map definition.
However, **PICIN** does not validate the input, so it should not be coded.
[**PICIN** and **PICOUT** are not valid for **ASSEMBLER** programs.]

COBOL allows multiple **CURRENCY SIGN** clauses, with a new **PICTURE SYMBOL** phrase that can define a symbol of one or more characters. For example:

SPECIAL-NAMES.

```

CURRENCY SIGN IS '$' WITH PICTURE SYMBOL '$'.
CURRENCY SIGN IS '£' WITH PICTURE SYMBOL '£'.
CURRENCY SIGN IS '€' WITH PICTURE SYMBOL '€'.
CURRENCY SIGN IS 'SFR' WITH PICTURE SYMBOL '#'.

```

WORKING-STORAGE SECTION.

```

01 USPRICE PIC $99.99.
01 UKPRICE PIC £99.99.
01 ECPRICE PIC €99.99.
01 CHPRICE PIC #99.99.

```

PROCEDURE DIVISION.

```

MOVE 12.34 to UKPRICE. value is £12.34
MOVE 12.34 to USPRICE. value is $12.34
MOVE 12.34 to ECPRICE. value is €12.34
MOVE 12.34 to CHPRICE. value is SFR12.34

```

The **DFHMDF** macro supports **PICIN** and **PICOUT** picture specifications with currency symbols other than \$. Previously it defaulted to the national currency symbol. You must use **LENGTH** when **PICOUT** specifies a COBOL picture containing a currency symbol that will be replaced by a currency sign of length greater than 1.

The symbolic map

This example shows the symbolic map generated from the following BMS macros:

```

MMPA1      DFHMSD TYPE=DSECT,LANG=COBOL,MODE=INOUT,TIOAPFX=YES
MMPAM01    DFHMDI SIZE=(24,80)
           DFHMDF POS=(1,17),LENGTH=18,                      X
             INITIAL='EMPLOYEE SELECTION',ATTRB=(ASKIP)
           DFHMDF POS=(8,3),LENGTH=11,                       X
             INITIAL='KEY EMPNO :',ATTRB=(ASKIP)
EMPNO      DFHMDF POS=(8,16),LENGTH=6,                      X
             INITIAL=' ',ATTRB=(UNPROT,NUM,BRT,IC)
           DFHMDF POS=(8,23),LENGTH=1,                      X
             INITIAL=' ',ATTRB=(ASKIP)
           DFHMSD TYPE=FINAL
           END

```

```

01  MMPAM01I.
    02  FILLER          PIC X(12).
    02  EMPNOL          PIC S9(4)  COMP.
    02  EMPNOA          PIC X.
    02  FILLER          REDEFINES EMPNOA.
      03  EMPNOF        PIC X.
    02  EMPNOI          PIC 9(6).
01  MMPAM01O          REDEFINES MMPAM01I.
    02  FILLER          PIC X(12).
    02  FILLER          PIC XXX.
    02  EMPNOO          PIC 9(6).

```

Had the language parameter on the DFHMSD been PLI, then the map would be:

```

DECLARE 1  MMPAM01I AUTOMATIC UNALIGNED,
        2  DFHMS1          CHARACTER (12),
        2  EMPNOL          FIXED BINARY (15,0),
        2  EMPNOF          CHARACTER (1),
        2  EMPNOI          CHARACTER (6);
DECLARE 1  MMPAM01O BASED(ADDR(MMPAM01I)) UNALIGNED,
        2  DFHMS2          CHARACTER (12),
        2  DFHMS3          FIXED BINARY (15,0),
        2  EMPNOA          CHARACTER (1),
        2  EMPNOO          CHARACTER (6);
/* END OF MAP DEFINITION */

```

Notice that the mapname MMPAM01 has been suffixed in the DSECT by

I - for the input map

O - for the output map

(A *DSECT* is a map of storage; this is used extensively in Assembler and is similar to a COBOL record definition or copybook. You can redefine this with your own record in order to get round the problem of wanting to describe GROUP entries on OCCURS fields.)

These suffixes also apply to the data field **EMPNO** i.e. EMPNOI and EMPNOO.

The additional fields in the symbolic map are suffixed by L, A, and F. They are:

EMPNOL on input, the length of data keyed into the field. If set to -1 on output, this determines the cursor position.

EMPNOF a 'flag' field normally set to nulls on input, unless the field has been modified but no data keyed - in which case it is X'80'.

EMPNOA redefining the flag field. On output, an attribute byte value may be placed here overriding the map definition.

The above fields may be checked and manipulated in an application program in response to operator action. Attributes and data may be altered and sent back to the terminal with any default information being retrieved from the physical map definition.

Also:

EMPNOC color of the field

EMPNOH highlight option for the field

Standard attribute and printer control character list, DFHBMSCA

Constant	Meaning
DFHBMPPEM	Printer end-of-message
DFHBMPNL	Printer new-line
DFHBMPFF	Printer form feed
DFHBMPCR	Printer carriage return
DFHBMASK	Autoskip
DFHBMUNP	Unprotected
DFHBMUNN	Unprotected and numeric
DFHBMPRO	Protected
DFHBMBRY	Bright
DFHBMDAR	Dark
DFHBMFSE	MDT set
DFHBMPRF	Protected and MDT set
DFHBMASF	Autoskip and MDT set
DFHBMASB	Autoskip and bright
DFHBMPSO	shift-out value X'0E'.
DFHBMPSI	shift-in value X'0F'.
DFHBMEOF	Field erased
DFHBMCUR	Field containing cursor flagged
DFHBMEC	Erased field containing cursor (COBOL only)
DFHBMFLG	Flags (COBOL only)
DFHBMDET	Field detected
DFHSA	Set attribute (SA) order
DFHERROR	Error code
DFHCOLOR	Color
DFHPS	Programmed symbols
DFHHLT	Highlight
DFH3270	Base 3270 field attribute
DFHVAL	Validation
DFHOUTLN	Field outlining attribute code
DFHBKTRN	Background transparency attribute code
DFHALL	Reset all to defaults
DFHDFT	Default
DFHDFCOL	Default color
DFHBLUE	Blue
DFHRED	Red
DFHPINK	Pink
DFHGREEN	Green
DFHTURQ	Turquoise
DFHYELLO	Yellow
DFHNEUTR	Neutral
DFHBASE	Base programmed symbols
DFHDFHI	Normal
DFHBLINK	Blink
DFHREVRS	Reverse video
DFHUNDLN	Underscore
DFHMFIL	Mandatory fill
DFHMENT	Mandatory enter
DFHMFET	Mandatory fill and mandatory enter
DFHMT	Trigger
DFHMFT	Mandatory fill and trigger
DFHMET	Mandatory enter and trigger
DFHMFET	Mandatory fill and mandatory enter and trigger
DFHUNNOD	Unprotected, nondisplay, nonprint, nondetectable, MDT
DFHUNIMD	Unprotected, intensify, light-pen detectable, MDT

COPY DFHBMSCA.

Put this in programs which handle screens.

DFHUNNUM	Unprotected, numeric, MDT
DFHUNNUB	Unprotected, numeric, intensify, light-pen detectable
DFHUNINT	Unprotected, numeric, intensify, light-pen detectable, MDT
DFHUNNON	Unprotected, numeric, nondisplay, nonprint, nondetectable, MDT
DFHPROTI	Protected, intensify, light-pen detectable
DFHPROTN	Protected, nondisplay, nonprint, nondetectable
DFHDFFR	Default outline
DFHUNDER	Underline
DFHRIGHT	Right vertical line
DFHOVER	Overline
DFHLEFT	Left vertical line
DFHBOX	Underline and right vertical and overline and left vertical
DFHSOSI	SOSI=yes
DFHTRANS	Background transparency
DFHOPAQ	No background transparency

Many installations have their own copybook of useful attribute byte combinations.

COPY ATTRIBUT.

```

BROWSE      OAK9FT.IFBA.BASE.CPY(ATTRIBUT) - 02.01      Line 00000000 C
Command ==>                                         Scroll
***** Top of Data *****
02  ATTRIBUTES-WS.
03  NORMAL-UNPROT-MDTOFF          PIC X VALUE ' '.
03  NORMAL-UNPROT-MDTON          PIC X VALUE 'A'.
03  HIGHLIGHT-UNPROT-MDTOFF      PIC X VALUE 'H'.
03  HIGHLIGHT-UNPROT-MDTON      PIC X VALUE 'I'.
03  DARKEN-UNPROT-MDTOFF        PIC X VALUE '<'.
03  DARKEN-UNPROT-MDTON        PIC X VALUE '('.
03  NORMAL-PROT-MDTOFF          PIC X VALUE '-'.
03  NORMAL-PROT-MDTON          PIC X VALUE '/'.
03  HIGHLIGHT-PROT-MDTOFF      PIC X VALUE 'Y'.
03  HIGHLIGHT-PROT-MDTON      PIC X VALUE 'Z'.
03  DARKEN-PROT-MDTOFF        PIC X VALUE '%'.
03  DARKEN-PROT-MDTON        PIC X VALUE '_'.
03  NORMAL-ASKIP-MDTOFF        PIC X VALUE '0'.
03  NORMAL-ASKIP-MDTON        PIC X VALUE '1'.
03  HIGHLIGHT-ASKIP-MDTOFF     PIC X VALUE '8'.
03  HIGHLIGHT-ASKIP-MDTON     PIC X VALUE '9'.
03  DARKEN-ASKIP-MDTOFF      PIC X VALUE '@'.
03  DARKEN-ASKIP-MDTON      PIC X VALUE QUOTE.

```