FHS V8.3A

Format Handling System for openUTM, TIAM, DCAM

Edition October 2006

Comments… Suggestions… Corrections…

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Fax forms for sending us your comments are included at the back of the manual.

There you will also find the addresses of the relevant User Documentation Department.

Certified documentation according to DIN EN ISO 9001:2000

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2000.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright and Trademarks

Copyright © Fujitsu Siemens Computers GmbH 2006.

All rights reserved.
Delivery subject to availability; right of technical modifications reserved.

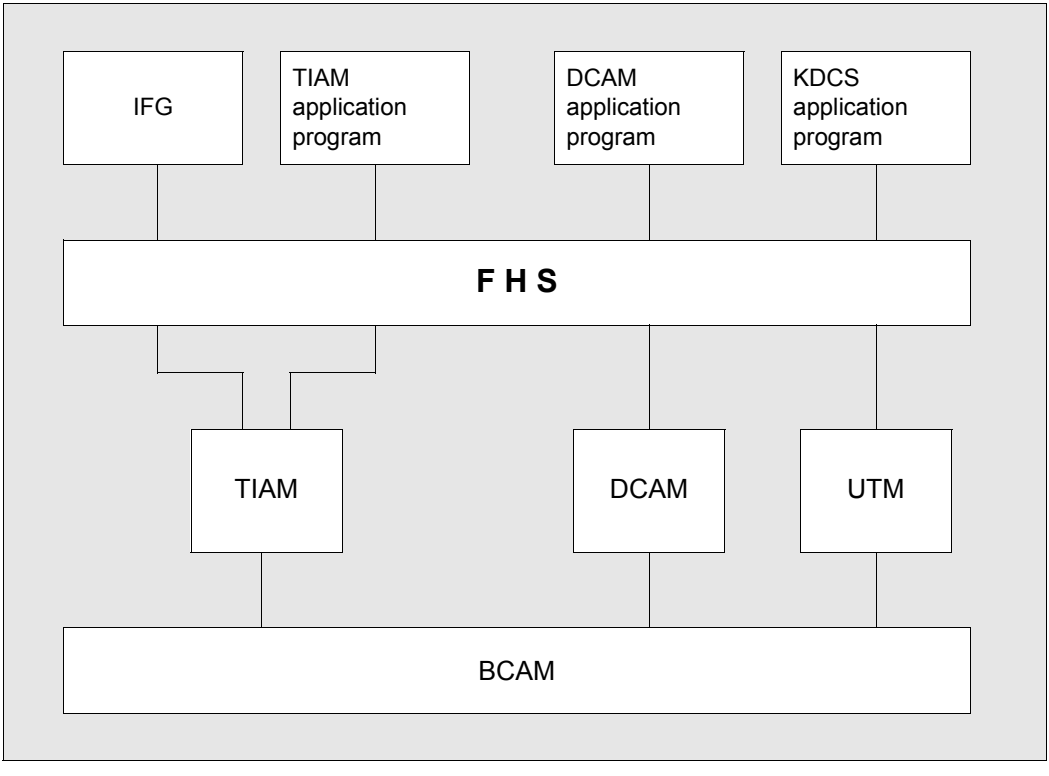
All hardware and software names used are trademarks of their respective manufacturers.

Contents

Contents

1 Preface

The **F**ormat **H**andling **S**ystem (FHS) supports the exchange of formatted messages between application programs and terminals. The use of FHS makes the application program largely independent of the physical characteristics of the terminals. FHS can be used for application programs in inquiry and transaction processing and in the timesharing operation. The following figure illustrates the integration of FHS in the system environment.



FHS in the system environment

FHS operates with formats prepared in advance using the Interactive Format Generator (IFG).

FHS is documented in the following manuals:

– FHS - Formatting System for openUTM, TIAM, DCAM

– FHS - Dialog Extension for TIAM and SDF-P

1.1 Target group

This User Guide describes the functions, application, and program interfaces of FHS. It is intended for terminal users and programmers who use one of the interfaces for remote processing in BS2000 (TIAM/RTIO, DCAM, openUTM).

In order to understand this manual, a basic knowledge of the BS2000 operating system and of the programming language being used is required.

The use of the FHS dialog manager (FHS-DM) is described in the FHS manual “Dialog Extension for TIAM and SDF-P”. It is intended for terminal users and programmers who use the TIAM interfaces for remote processing in BS2000.

1.2 Summary of contents

This manual is divided into the following chapters:

- Introduction
- Purpose, mode of operation and functions of FHS and also the various data transfer areas. These chapters are largely independent of the application and of the programming language and are therefore important to all users.
- The use of FHS in UTM applications

  This chapter contains the essential FHS-specific information required by the UTM user in order to be able to use a UTM application with FHS.

  The chapter entitled “FHS dialog extension” applies only to UTM applications in openUTM V3.3 and later.
- The use of FHS in ASSEMBLER and COBOL programs for DCAM/TIAM users

  These two chapters show the structure of the programs and describe the requisite macros and the supply of data to the data structures.

- The use of FHS in Fortran and PLI programs for TIAM users

  These two chapters show the structure of the programs and describe the supply of data to the data structures.

- FHS utility routines and print routines.
- The appendix contains examples of addressing aids, return codes and their significance, device-related information and tables.

Before you write an application, you should at least read the chapters about the purpose, mode of operation and functions of FHS and about the various data transfer areas. After this you should read one of the application-specific chapters.

1.3 Changes since the last version of the manual

Unicode formats

The exchange of formatted messages containing Unicode strings between application programs and terminals is supported. This is made possible by the use of Unicode formats generated by IFG V8.3A or up.

1.4 Changes since the version 8.1A

– **Loading of formats from different format libraries**

  Additional format libraries can be assigned on the basis of a link name without a new interface having to be specified or a new start parameter analysis having to be performed.

– **Service function ‘Unload Format’**

  This function allows formats to be unloaded so that they can be replaced by modified formats without having to unload the application.

– **Service function ‘Dynamic Retrieval of Information on the Structure of the Addressing Aid for #Formats’**

1.5 README file

Information on any functional changes and additions to the current product version described in this manual can be found in the product-specific README file. You will find the README file on your BS2000 computer under the file name SYSRME.FHS.083.E.
The user ID under which the README file is cataloged can be obtained from your system administrator. You can view the README file using the /SHOW-FILE command or an editor, and print it out on a standard printer using the following command:

```
PRINT-FILE FILE-NAME=filename,LAYOUT-CONTROL=PARAMETERS(CONTROL-
CHARACTERS=EBCDIC)
```

2 Introduction to FHS

What is a format?

A format (also known as a mask or, formerly, map) is a form displayed on the screen of a data display terminal. Just like the forms we meet every day (e.g. application forms, order forms), a format consists of fields (boxes) in which entries can be made, and predefined texts which are part of the form itself. Such a “form” is based on a logical data structure made up of:

– fields with fixed text (text fields)

– fields in which entries can be made by the terminal user and/or the application program (variable fields)

– information about the position of these fields on the screen

– information about the characteristics of the format (e.g. the terminals on which the format can be output)

– information about the characteristics (attributes) of the fields of the format (e.g. underline)

– information about the editing characteristics of field contents.

Unicode formats are formats in which the UNICODE field attribute has been assigned at least to one field of a format in IFG or formats for which the UNICODE attribute has been forced, even if this format does not contain any UNICODE field.

The consequence of assigning the UNICODE attribute to a field is that:

– the user is able to type any character from the Unicode Basic Multilanguage Plane (BMP) into this field

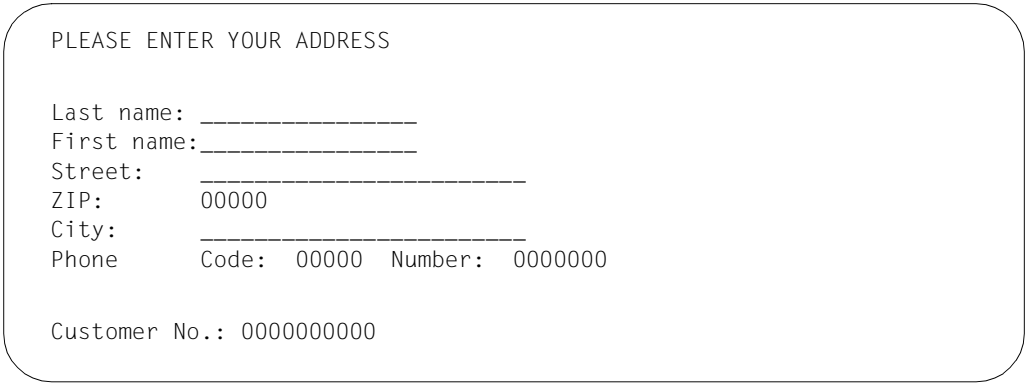
– the application program will receive the content of the field in the addressing aids in an area having a size of 2 bytes per character.

– the encoding of this field in the application program is UTF-16.

For more information concerning the way to assign the UNICODE attribute to a field or to a format, please refer to the “IFG for FHS” manual.

Refer to the „Unicode in BS2000/OSD” introduction for a survey of the Unicode support in the BS2000/OSD as well as basic information on Unicode.

The following figure shows an example of a format as it is displayed on the screen.



Example of a format

Since FHS makes the application programs independent of the physical characteristics of terminals, the user can work with different terminals without having to be familiar with their varying physical characteristics. The user works with virtual terminals and FHS provides the interface to the actual terminals.

Which terminals does FHS work with?

FHS V8.3 supports operation with

– 8160, 9750, 9755, 9763 Data Display Terminals, 3270 display terminals, and also equivalent devices and emulations

– 9001, 9002, 9003, 9004, 9011, 9012, 9013 and 9022 printer terminals, PCL printers (9021, 9022-200) and 3287 printers.

The 3287 printer must be connected to an 8112 control unit. Formats for this printer may not be generated for ’fast formatting’.

For FHS to support the IBM System 3270, the software product TRANSIT-CD must be installed in the front-end processor and the terminals must be generated as system device type 3270.

The printer terminals can be connected either locally to a data display terminal or centrally via a printer terminal controller.

If the terminal type is specified incorrectly in the PDN, errors can occur during formatting. The actual terminal type and the terminal type generated in the PDN must be identical.

The Unicode support is available with MT9750 (Windows) V6.0B/V6.1 or compatible emulations.

At the user interface, these terminal emulations support two ways of working:

– Either the whole format is in Unicode mode, that means that you are allowed to type any character from the Unicode BMP corresponding to the characters U+000000 to U+00FFFF into each input field.

– Or the format on the screen is in a non-Unicode mode and therefore the character set allowed for entering texts in input fields is restricted to a 7-bit or 8-bit (ISO8859-x) character set.

A mix of these two modes is not allowed: The emulation cannot restrict one input field to ISO8859-1 while you may enter any character of the Unicode BMP in another field of the same format – and thus if the whole format is UNICODE. If the characters entered in the non-UNICODE field are not compatible with the base coded character set of the format, FHS will generate a return code.

Refer to the “Unicode in BS2000/OSD” introduction for a survey of the Unicode support in the BS2000/OSD as well as basic information on Unicode.

Programming languages

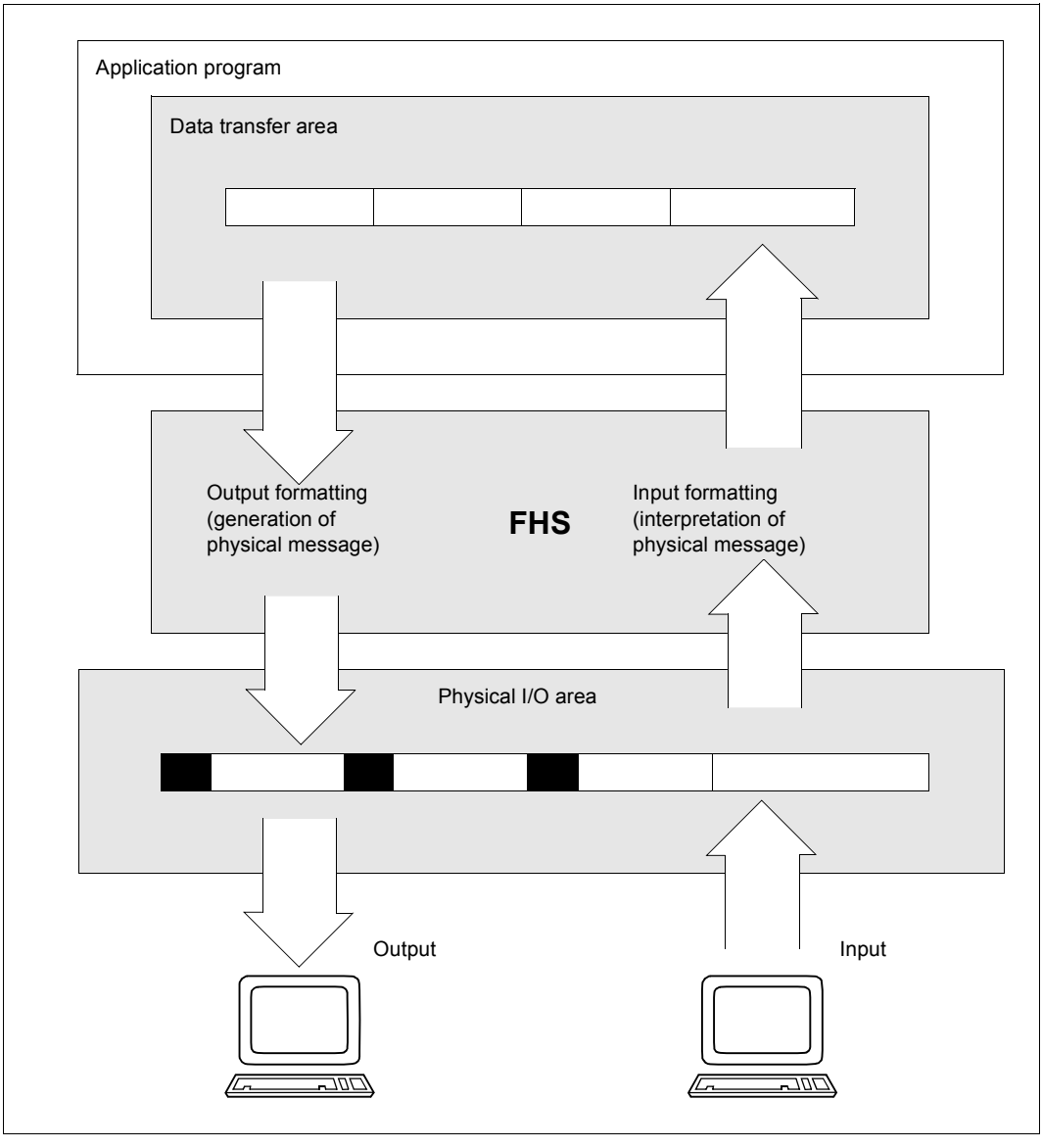
The use of FHS in a TIAM/DCAM environment with ASSEMBLER and COBOL is described in this manual. The formats generated using IFG can also be used in other programming languages, e.g. PL/I, Pascal, Fortran etc. Use of FHS in UTM applications is described in the appropriate openUTM manuals. In TIAM/DCAM applications, FHS can be used for PL/I, Pascal and Fortran in a similar manner as for COBOL (see page 297). The interface structures in the individual programming languages are provided or must be created by the user. Application options using RPG are described in the “RPG3 (BS2000)” User Guide.

Use on XS systems

FHS runs in the same address area that also contains the application program. FHS can thus only utilize the upper address area (> 16 Mbytes) if the application program has also been loaded into the same address area.

How FHS functions

FHS supports the input and output of formatted messages that are exchanged interactively between application program and terminal. The following figure shows the processes of message input and message output.



Message input and output

The application program supplies the data to the output data transfer area. FHS adds terminal-specific control characters and also fixed texts defined in the format and assembles a complete output message in the physical input/output area. From this area the output message is sent to the terminal by means of a call appropriate to the access method. The terminal user can then deal with the displayed format.

When the format has been processed on the screen, transfer to the processor can be initiated. A call appropriate to the access method is issued to transfer the message to the physical input/output area. FHS removes the control characters contained in the message and makes available to the application program the variable fields in the input data transfer area. The application program can now process the data. FHS provides additional information about the formatting operation in the form of return codes and acknowledgments.

The same data transfer area may be used for both input and output, or different areas can also be used.

The characteristics of the format and of the individual fields in the format are defined by assigning attributes. Attributes are assigned during format generation using IFG (static attributes) or in the application program by way of the global and field attributes (dynamic attributes).

Loading FHS

FHS uses a formatting routine that is invisible to the user; this comprises a number of modules that are dynamically loaded by FHS when required. Only those modules that are actually required for formatting are loaded.

The FHS modules are loaded from the file assigned by means of the command

```
/SET-FILE-LINK LINK=MROUTLIB,FILE-NAME=libraryname
```

If no such FILE assignment has been made, loading takes place from the system file TASKLIB. The latter can be assigned by means of the command

```
/SET-TASKLIB LIBRARY=libraryname
```

If no assignment has been made, an attempt is made to load the formatting routine from the user file TASKLIB, and finally from the file $.TASKLIB.

The FHS modules can also be linked statically to the application program (using the linkage editor BINDER). FHS reloads its modules again itself unless the dynamic linking loader DLL is informed by the user through the system macro TABLE that FHS is already present. Static linking-in of FHS should thus only be done in conjunction with the system macro TABLE since otherwise only the dynamically loaded FHS modules are significant, and not the statically linked-in FHS modules. See the BS2000/OSD-BC manual “Executive Macros” for a description of the TABLE macro.

If all FHS modules are linked statically, then the following INCLUDE statements must be specified in the BINDER procedure:

```
//INCLUDE-MODULE E=(MFHSROUT),T=R,LIB=mroutlib
//INCLUDE-MODULE E=(MFHSDC4C),T=R,LIB=mroutlib
//INCLUDE-MODULE E=(MFHSCTAB),T=R,LIB=mroutlib
```

Loading the formats

The formats are loaded from the file assigned by means of the command

```
/SET-FILE-LINK LINK=MAPLIB,FILE-NAME=libraryname
```

If no such assignment was made, an attempt is made to load from the file F.MAPLIB.

In the case of UTM applications (UTM ≥ 3.1), the format library is assigned by means of the UTM start parameter card

```
.FHS MAPLIB=libraryname
```

(default here is also F.MAPLIB).

Additional format libraries can be assigned on the basis of the link name BLSLIBnn (nn = a value between 00 and 99). Format libraries can be ordered hierarchically using the number of the BLSLIBs without a new interface having to be specified or a new start parameter analysis having to be performed. This functionality can be applied directly in a TIAM/DCAM application with the assistance of the FHS kernel.

Formats are then searched for in the following sequence:

First, the library last used is searched. Then the libraries that were assigned to the link name. The ascending order of the BLSLIBs numbering is taken into account here.

Because the MAPLIB start parameter is retained in the openUTM environment, this interface is fully compatible with the previous version.

3 FHS functions

3.1 Format types

The individual format types differ in the structure of the data transfer area. The structure of the data transfer area and thus of the format type is defined during format generation using IFG. The following types of format exist:

– #formats:

  This format type uses the data transfer area with separate attribute blocks and field contents. This data transfer area allows utilization of all the functions of FHS V6.0 or later.

– +formats:

  This format type uses the data transfer area with attribute fields.

– *formats:

  This format type uses the data transfer area without attribute fields.

You will find further details on the data transfer areas in chapter “Structure of the data transfer area” on page 45.

3.2 Outputting formats

In order to output a format on a data display terminal, the data to be incorporated into the format must first be placed in the data transfer area. FHS then constructs a complete output message in the physical input/output area, adding to the format data terminal-specific control characters and fixed texts defined in the format. FHS then sends the entire message to the terminal.

If one field of a format is designed to receive Unicode characters, then the whole format is output in Unicode mode and thus the user will be able to type Unicode strings in all input fields of the format.

FHS will analyze the input of the user and convert this text according to the definition of the corresponding field in the format:

– If the field is declared as a UNICODE field, FHS will check and edit the field and then convert it to UTF-16 and transfer it to the addressing aids of the format.

– If the field is not declared as a UNICODE field, the FHS will check that all characters entered in this field are compliant with the base character set of the format – as defined in IFG – or the character set defined for this USER/LTERM in UTM or in the FHS CONTROL BLOCK (same processing as for the support of 8-bit formats in FHS).

  If one character is not compatible with the base character set, then FHS will follow the existing rules for the field validation check failures, i.e. FHS will

  – either issue a return code in the addressing aids in the EDIT_RC part of the field attribute block corresponding to this field and in the FIELDS VALIDATION part of the global attribute block (FIELDS VALIDATION INVALID) if FHS-DE is not enabled.

  – or issue a FHS-DE standard error message “At least one character is incompatible with the base character set of the format”. This message follows the usual rules for the FHS-DE standard error messages: it can be easily translated by the application developer depending on the language of the application.

Refer to the „Unicode in BS2000/OSD“ introduction for a survey of the Unicode support in the BS2000/OSD as well as basic information on Unicode.

A distinction is made between the output of a new format (new output) and the output of an updated format (update output).

New output

When a new format is output on a data display terminal, the screen is first cleared and then the complete format is displayed as generated by IFG, including the fields that contain fixed text. The variable fields are displayed either as filled by the application program or containing the specified output fill characters. The format is output in its entirety, regardless of the previous appearance of the screen, i.e. the area of the screen occupied by the format is overwritten completely.

Update output

In update output, only parts of the format already displayed on the screen are modified. Only parts that are accessible to the program can be updated (field attributes, field contents). The application program must flag these parts before output formatting takes place. This means:

– with #formats:

  only those parts of the format that are to be updated on the screen also need to be modified in the application program. This is known as the differential output.

– with all other format types:

  all those parts of the format that are not to be redisplayed must be cleared to X'00...00'.

3.3 Screen restart

FHS requires a restart area of sufficient length for the screen restart facility. The contents of this area allow FHS to reconstruct the screen at any time in such a way that the last completely formatted screen is displayed.

This restart area must be made available by the application program for DCAM COBOL applications and for TIAM/DCAM ASSEMBLER applications. This area is automatically made available for UTM and TIAM COBOL applications.

This restart area must always be present for #formats; it is optional for other format types.

3.4 Data editing

FHS can edit and check the field contents of a format in accordance with certain defaults. How the field contents are to be edited and checked is defined during format generation using IFG. Some editing characteristics apply to the entire format, others can be defined for individual fields (see the manual “IFG for FHS”).

You can only utilize all the data editing facilities if the data transfer area with separate attribute blocks and field contents is used (#formats). FHS also checks the field contents in this case. The result of this check can be obtained from the global attribute ’Fields Validation’ and the field attribute ’Edit State’. If errors are detected during field processing, FHS provides a field-specific return code in the field attribute ’Edit-RC’.

Formats that do not use this data transfer area can only utilize the function ’Field Alignment and Fill Characters’.

When the data editing functions concern UNICODE fields, all special characters specified in IFG for this field – like the input and output fill characters – are translated to their UTF-16 corresponding code and handled in this encoding.
For example, if you specify the EBCDIC character space (X'40') as the output fill character, then the UTF-16 hexadecimal value X'0020' corresponding to the space character will not be transferred to the terminal if it is present at the beginning or at the end of an UTF string in the addressing aids.

3.4.1 Field alignment and fill characters

You can decide to have the fields in your format aligned during input and/or output, and whether you wish to have unoccupied positions filled with a fill character.

Options:

– no justification
– left justification
– right justification

You define during format generation with IFG how the fields are to be aligned and what fill characters are to be used.

Note

– Which characters in the send field count as part of the character string depends on the fill characters specified. Normally the character string begins with the first printable character and ends with the last printable character in the send field.

– Within a character string non-printable characters are replaced by the SUB (X'3F') character; NUL characters stay as they are.

– If there are no printable characters in the send field, the receive field is filled with fill characters (the effective length is then 0).

– Characters that are the same as the fill character (or NULL or non-printable), and those at the beginning and end of a character string, are not transferred. "Transferred" means that FHS passes the characters to the application program's transfer area in input formatting and from the transfer area to the input/output area during output formatting.

  Note that

  – in output formatting only the fill characters for output formatting are not transferred.

  – in input formatting the fill characters for input formatting and output formatting are not transferred.

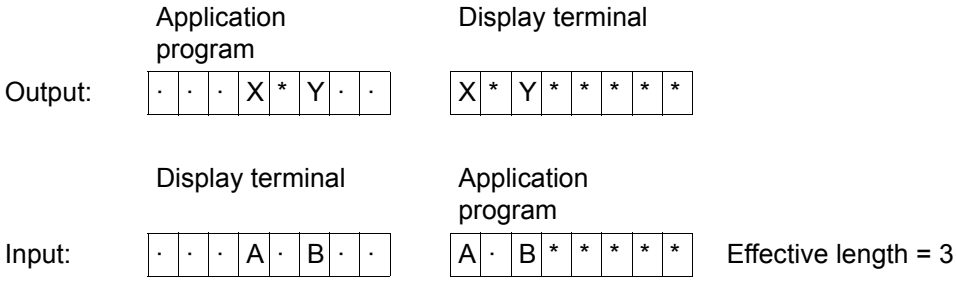
  – the remainder of the receive field is filled with fill characters for the relevant transfer direction.

  There is a special rule for zeros because they are transferred in input formatting even if zero is the input fill character. If zero is only an output fill character, zeros are treated like any other character and not transferred during input formatting.

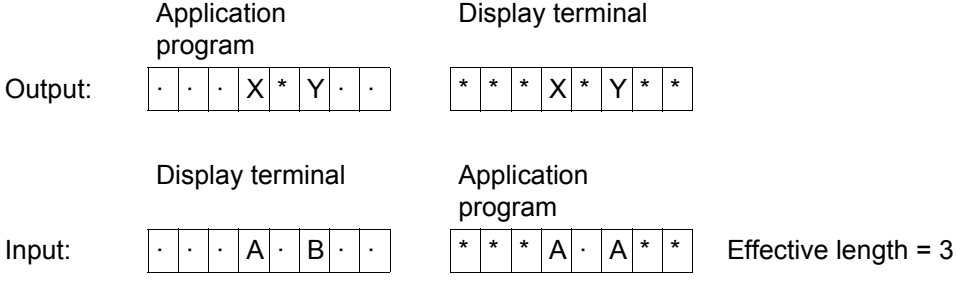
– The effective field length in the associated length field after input formatting is equal to the number of characters passed to the application program's transfer area or half of the number of bytes in case of a UNICODE field. This is the number of characters in the field without the fill characters at the beginning or end; if zero is the fill character for input formatting, only the relevant zeros, not the fill character zeros, are included in the effective length.

Example of how character strings are aligned (the character . indicates a NULL character):

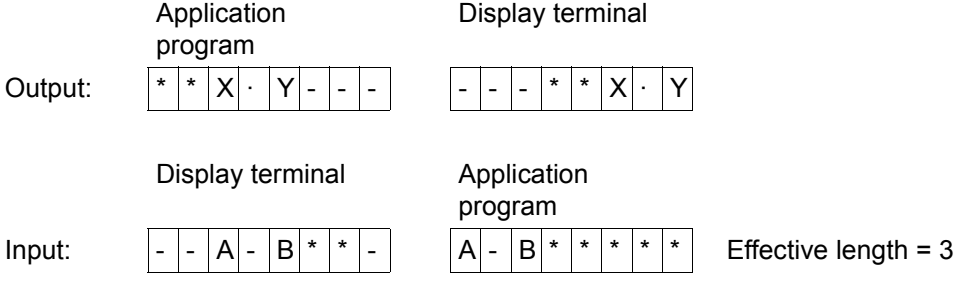
1. `Justification` for input and output formatting: left
   `Fill character` for input and output formatting: *



2. `Justification` for input and output formatting: none
   `Fill character` for input and output formatting: *



3. `Justification` for input formatting: left
   Justification for output formatting: right
   `Fill character` for input formatting: *
   Fill character for output formatting: -



4. `Justification` for input and output formatting: right
   `Fill character` for input and output formatting: 0
   Field data type not arithmetic

	Application program	Display terminal
Output:	· · 1 0 , 0 0 ·	0 0 0 1 0 , 0 0

	Display terminal	Application program	
Input:	· · 2 0 , 0 0 ·	0 0 0 2 0 , 0 0	Effective length = 5

5. `Justification` for input formatting: none
   Justification for output formatting: right
   `Fill character` for input formatting: * (≠ 0!)
   Fill character for output formatting: 0

	Application program	Display terminal
Output:	· 0 0 1 0 0 · ·	0 0 0 0 0 1 0 0

	Display terminal	Application program	
Input:	· 0 0 1 0 1 0 0	* * * 1 0 1 * *	Effective length = 3

	Application program	Display terminal
Output:	* 0 1 0 , 0 * ·	0 * 0 1 0 , 0 *

	Display terminal	Application program	
Input:	· * 0 2 2 0 * ·	* * * 2 2 * * *	Effective length = 2

3.4.2 Editing and checking functions

The functions described in this section can only be used with #formats.

Note

The attributes „field data type“, „decimal separator“, „digit separator“, „number of decimal places“, „zero suppression“, „sign permitted“, „digit grouping“, „date format“ are incompatible with the Unicode implementation and are thus mutually exclusive in IFG.

That means, for example, that the only permissible characters for arithmetic fields remain the digits 0 through 9 and, in certain positions, depending on the decimal separator, digit separator and sign attributes, the characters ',', '.', ' ', '+' and '-'. No other character will be allowed and the internal representation of these fields in the addressing aids will be the same as for other non-UNICODE fields: 1 byte per character.

Field data type

There are four different data types:

– any desired characters;
the field may contain any printable characters.

– alphabetic;
the field may contain only letters (’A’ through ’Z’ and ’a’ through ’z’) and blanks.

– arithmetic;
the only permissible input characters for this field are the digits 0 through 9 and in certain positions, depending on the decimal separator, digit separator and sign attributes, the characters ’,’, ’.’, ’_’, ’+’ and ’-’. Blanks followed by output fill characters before the number or output fill characters followed by blanks after the number will result in an error if the output fill character is non-blank. No arithmetic field may contain more than 15 digit positions. The number of digit positions is given by deducting from the field length the number of positions for signs, decimal separators and digit separators, where the corresponding attributes have been set. The application program may only supply arithmetic fields with digits, followed on the right by a sign where appropriate. When an arithmetic field with no relevant contents is output, output fill characters appear in the field. No editing takes place in this field. If an arithmetic field is deleted on the screen and if nothing is entered in this field, then “0” and, if defined, a positive sign are placed in the data transfer area. The field attribute ’Input State’ will indicate that the field has been deleted and no relevant characters were entered. This function makes it possible to distinguish unknown values from the value “0”.

– date;
this field may contain only the digits 0 through 9, blanks and two occurrences of the same separator. Blanks followed by output fill characters before the date or output fill characters followed by blanks after the date will result in an error if the output fill character is non-blank.

FHS checks on input and output whether the field contains only characters of the appropriate type with the relevant length.

Minimum input length

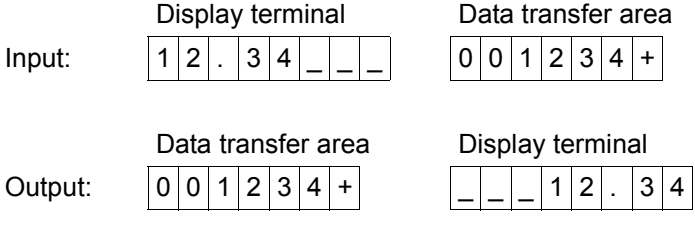
FHS checks on input whether a character string was entered with the specified minimum input length. No check is made if nothing was entered. 'Mandatory Input' should therefore always be defined for such fields during format generation using IFG.

Decimal separator

FHS checks on input whether decimal places are permitted and whether the specified decimal separator character occurs only once in the field. FHS removes the separator character.

Example

Decimal separator: '.'; decimal places: 2
sign: permitted and floating on the left



Digit separator

FHS checks on input whether only the defined character occurs as digit separator at specific positions in the field. FHS removes the separator character.

Example

Digit separator: ’,’
decimal separator: ’.’
decimal places: two; sign: permitted and floating on the left

	Display terminal	Data transfer area
Input:·	1 , 2 3 4 · · · ·	1 2 3 4 0 0 +

	Data transfer area	Display terminal
Output:	1 2 3 4 0 0 +	_ 1 , 2 3 4 . 0 0

Number of decimal places

FHS checks on arithmetic fields to insure that the number of decimal places entered after the decimal separator does not exceed the specified number of decimal places. Decimal places not entered are right-filled with ’0’. It is not permissible to enter more than the specified number of decimal places. On output, FHS inserts the decimal separators. Output is right-justified.

Zero suppression

This attribute is permitted only for arithmetic fields.

On input, leading zeros before the decimal separator may be omitted, e.g. it is possible to enter ’.52’.

On output, all leading zeros except for the decimal units position can be replaced by output fill characters. Only when no further position is available before the decimal separator is there no further digit before the decimal separator.

Sign permitted

This attribute is permitted only for arithmetic fields.

When a sign is permitted for a field, the sign may be entered at the left or at the right. If no sign is permitted, FHS checks on input to insure that neither the ’+’ sign nor the ’–’ sign is present in the field.

On input of a negative number, FHS places a negative sign in the last position of the field in the data transfer area; a positive sign or no sign on input results in a ’+’ sign in the last position of the field in the data transfer area. Zero is always regarded as a positive number.

On output, a positive sign is represented as a blank and a negative sign as ’–’. Depending on the ’Floating Sign’ attribute, the sign is placed either in the last position in the field or before the number. The remaining positions to the left are filled either with output fill characters or, in the case of ’Floating Sign’, with blanks.

Floating sign

This attribute is permitted only for arithmetic fields and only in conjunction with the ’Sign Permitted’ attribute.
On input, the sign may stand to the left or right of the entered number.

On output, the number is transferred to the field in accordance with the leading positions and decimal places. A negative sign is placed before the number as ’–’. The positions to the left of the sign are filled with blanks.

Digit grouping

This attribute is permitted only for arithmetic fields.

The digit separator does not need to be entered on input. If it is entered, FHS checks whether the leading positions are separated from right to left in groups of 3 by the digit separator.

On output, FHS inserts the digit separator in the correct position; the leading positions are transferred, starting from the right, and the digit separator is inserted after every third digit, assuming there is at least one further digit. If ’Zero Suppression=NO’, the digit separator may also stand in the first position of the field.

Date format

On input, FHS checks the sequence of day, month and year, and whether the same separator appears in two positions. If the year is to be specified as a four-digit number, all four positions must be entered. During calendar checking the accounting day is computed and placed in the data transfer area.

In the case of a day, month and two-digit year specification, leading zeros can be omitted; blanks may stand before and after the separator.

On output, FHS checks whether the date is a valid Gregorian date if calendar checking was requested during format generation.

Note

Representation in the data transfer area *must* correspond to the representation of the GDATE macro for both input and output (ISO or ISO4 form); see the BS2000/OSD-BC manual “Executive Macros”. Only the on-screen representation can be specified with IFG (separators, sequence).

Example

Sequence: day,month,year
separator: /
year value: two-digit

	Display terminal	Data transfer area
Input:	1 / 4 / 8 8	8 8 - 0 4 - 0 1 0 9 2 ␣

The positions following the day in the data transfer area (here 092) signify the accounting day that is computed on the basis of the entered date when calendar checking is requested.

	Display terminal	Data transfer area
Input:	3 3 / 0 / 8 8	8 8 - 0 0 - 3 3 0 0 0 ␣
		(without calendar check)

	Data transfer area	Display terminal
Output:	8 8 - 0 4 - 0 1 · · · ·	0 1 / 0 4 / 8 8
		(with calendar check)
	8 8 - 0 0 - 3 3 · · · ·	3 3 / 0 0 / 8 8
		(without calendar check)

Day, month, year and separator are transferred to the appropriate positions of the field; the accounting day is not output and not checked. The positions for day, month and year are supplemented by leading zeros, if required. The output is thus always eight/ten positions long.
In order to delete the fields, the accounting day must also be set to NULL.

3.5 Undefined values

An undefined value exists if no relevant value (not even a zero or a blank) is entered for a field. During generation of a format with IFG, a substitute character for undefined values can be defined. This substitute character can be defined for each field type, and may be an alphanumeric character or a hexadecimal value.

Note

Undefined values may be used only for #formats.

During input formatting, FHS passes the following information in the data transfer area if the field is completely filled with the substitute character:

– the global attribute ’undefined values’ with the value ’UNDEFINED’ in the addressing aid to indicate that at least one field contains an undefined value;

– the value ’UNDEFINED’ for the field attributes ’input-state’ and ’input-state-actual’.

For output formatting, the field attribute ’output-control’ must be set to the value ’OUTPUT UNDEFINED’.

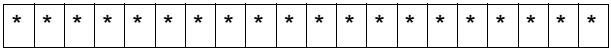
For all fields for which processing of undefined values is permitted, it is possible to specify whether the entire field or only the first position of the field must be filled with the substitute character. If only the first position of a field is filled with the substitute character and if ’fast detection’ was specified when the format was generated, processing of long fields will be considerably faster.

Example

Substitute character for undefined value: ’*’

*	S	T	D															

With ’fast detection’, this field receives the attribute ’undefined value’, since the first character is the substitute character for undefined values and only the first position of the field is checked. If, in contrast, ’fast detection’ was not selected for this field, the field does not receive the attribute ’undefined value’, since all positions of the field are checked for the substitute character.



In this case, the field always receives the attribute ’undefined value’, regardless of whether or not ’fast detection’ was specified.

The following should be noted when using undefined values:

– The fill character and the substitute character for undefined values may not be the same character.

– If, during input, the substitute character for undefined values is inadvertently entered, the field contents are irrelevant and are ignored.

– If an undefined value is entered for a field, the field attribute ’edit status’ is irrelevant, since there are no field contents and no editing is thus performed.

– If, after output of an undefined value for a field, no input is entered from the keyboard, and if the field does not have the attribute ’automatic input’, the field receives the input state ’NOT TOUCHED’ rather than ’UNDEFINED’. This means that FHS cannot check the values.

– A format with different field types may contain different substitute characters for undefined values.

– FHS rejects the undefined value if a conflict is possible (MODIFIED or CLEARED).

– If the field contents are undefined, no exit routine is called.

– When defining the substitute characters for undefined values in IFG, care must be taken that these values are permissible for the various fields; the characters ’A’ and ’!’, for example, are not permitted in a field with the attribute NUM-lock.

– A distinction is made between uppercase and lowercase letters for the substitute characters.

– In order to permit input of an undefined value to be detected, the field attribute ’output-control’ must be set to the value ’OUTPUT UNDEFINED’.

3.6 Partial formats

What are partial formats?

Partial formats are formats which use only a defined part of the terminal screen, thus making it possible for the program to construct a screen from more than one partial format. These partial formats must not overlap. The position of the individual partial formats on the screen is defined by means of the start line number upon format generation with IFG. With #formats, this position can be changed using the global attribute 'START LINE'. With +formats and *formats this position is fixed.

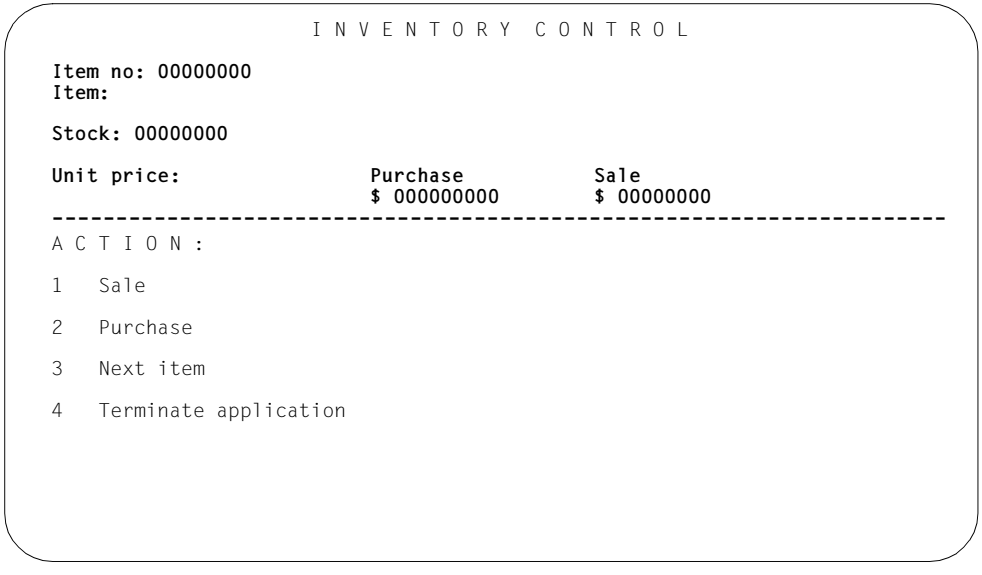
Partial formats can be swapped on the screen. If after exchanging a partial format there is overlapping with old partial formats already on the screen, FHS also removes what remains of these partial formats from the screen.

When at least one Unicode partial format is used in a partial formatting cycle, it is mandatory that all partial formats of the cycle are output in Unicode mode (see also section "Which terminals does FHS work with?" on page 18). The UNICODE format attribute can be used for this purpose in IFG.

Example of partial formats

Partial format 1 with the start line number 01 defined in IFG and partial formats 2a, 2b and 2c with the start line number 11 can be used as follows for example:

Partial formats 1 and 2a are first shown on the screen:



Depending on the action selected, partial format 2a only is replaced by 2b or 2c. Partial format 1 remains on the screen.

```
                    I N V E N T O R Y   C O N T R O L

Item no: 00000000
Item:

Stock: 00000000

Unit price:                Purchase           Sale
                        $ 000000000      $ 000000000
------------------------------------------------------------------------
A C T I O N :     S A L E

Customer no.:       000000000          Customer:
                                       Street:
                                       ZIP: 00000  City:

Quantity:           00000              Discount: 000 %
Unite price:     $ 0000000000          Total price: $ 0000000000
New Item (y/n):                        New Customer (y/n):
```

1

2b

```
                    I N V E N T O R Y   C O N T R O L

Item no: 00000000
Item:

Stock: 00000000

Unit price:                Purchase           Sale
                        $ 000000000      $ 000000000
------------------------------------------------------------------------
A C T I O N :     S A L E

Customer no.:       000000000          Customer:
                                       Street:
                                       ZIP: 00000  City:

Quantity:           00000              Discount: 000 %
Unite price:     $ 0000000000          Total price: $ 0000000000
New Item (y/n):
```

1

2c

More than one partial format can be sent to the terminal simultaneously. All partial formats output at the same time are formatted in one “partial formatting cycle”.

3.7 Checking data fields with an exit routine

You can use an exit routine to check the fields of a format for specific contents and to modify those contents. Exit routines are not standard software; they must be written by the user.

An exit routine is a routine separate from the application program. The exit routine is executed for a field only if this was specified in IFG and if the field is transferred. The exit routine is activated for this field during formatting after the fields have been edited by the formatting routine. The interface between FHS and the exit routine is the user exit interface (exit operand block).

Exit routines offer the following capabilities:

– they can be activated during any formatting operation,

– the fields can be handled in different ways, depending on their exit remark,

– the fields to be processed can be combined on the basis of their exit remark to form groups,

– the application program receives a return code that you can use in the exit routine,

– when changes are made in field handling, only the exit routine needs to be modified and recompiled.

3.8 Fast formatting

The basic idea of fast formatting is that the device protocol is contained within the format definition as far as possible. The result of this is that when the format is output, only parts of the message need to be generated dynamically. This reduces the required path length in FHS.

The disadvantage of this method is that these formats can only be output on a single device type and possibly related upward-compatible equipment (see table on page 536).

Notes

With partial formatting, it is not possible to change the device type within a partial formatting cycle and the following update outputs. If the partial formatting cycle is opened with a partial format for 'fast formatting', the device type specified in this partial format is applicable until a new partial formatting cycle is opened. This may restrict the capabilities of the actual terminal.

'Fast formatting' is not possible for the 3287 printer. In addition 'Fast formatting' is neither possible for DE-formats nor for Unicode formats.

For PCL printers, the character pitch cannot be changed if 'fast formatting' is used.

3.9 Service function

In addition to the formatting calls for input and output formatting, FHS also offers service functions for special services.

Three service functions are currently available:

- The service function 'Initialization of the Data Transfer Area with Separate Attribute Blocks and Field Contents' is currently available. This function sets all the field attributes in accordance with their default values in the format. The global attributes (apart from the global attributes for formatting acknowledgment) and the field contents remain unchanged. It is thus also possible at any time to reset to the initial status data transfer areas that have already been supplied with data. The way in which this function is called depends on the particular programming environment.
- 'Unload Format'. This function allows formats to be unloaded, for example for test purposes, so that they can be replaced by modified formats without having to unload the application.
- 'Dynamic Retrieval of Information on the Structure of the Addressing Aid for #Formats'. FHS returns information about the structure of the addressing aid for the current format.

3.10 Using different character sets

Formats for the 9763 Data Display Terminal can use different character sets created using the Interactive Character Set Editor ICE.

The assignment of the character sets to the fields of a format is effected during format generation with IFG. The field attribute that specifies the use of a character set in a field is defined during generation of the format. No facility is provided for dynamically updating this attribute.

Loading of the character sets into the data display terminal takes place when the format is output. The character sets are loaded from the format application file by FHS. When a character set that has already been loaded is used again, loading is suppressed. This facility requires the use of the connection-specific administrative area.

When using partial formats, you must insure that partial formats that may appear simultaneously on the screen are compatible with one another with respect to their character sets.

3.11 Loading P keys

The program key load and save system PLUS offers you the following options:

– as the terminal user, you can define interactively how the P keys are assigned,

– save such assignments as P programs (= P key formats) in format libraries and

– load them into the P keys of the Data Display Terminals.

– You can also have stored P programs loaded into the P keys by an application program. This normally occurs when FHS is used.

The full functional scope of PLUS and also the use of PLUS by the terminal user are described in the “TIAM (TRANSDATA, BS2000)“ User Guide.

P key formats using the PLUS control statement MSG should not be output to formatted screens.

P keys can be loaded each time a #format is output (integrated loading of P keys). This is described on page 48 under the global attributes ’Display Selection’ and ’P-Key Set’.

4 Structure of the data transfer area

The data transfer area structure to be used for the formats is defined during format generation with IFG. The following options are available:

– not justified, no attribute fields (*formats)

  In this instance, the field attributes defined during format generation are always applicable. They cannot be changed by FHS.

  Either all fields of *formats should be Unicode, or none of these fields should be able to contain Unicode characters. A mix of UNICODE and non-UNICODE fields is not recommended: If FHS detects that Unicode text is entered in a non-UNICODE field, it will issue the return code RC0424A4 that is global for the format but without being able to specify the concerned field. In these data structures, each character on a UNICODE field is represented by 2 bytes.

  Furthermore the application programmer is responsible for checking that the user only enters characters compatible with the EDF03 character set in a field used as UTM control field.

– justified, with attribute fields or

– not justified, with attribute fields (+formats)

  In this case, the field attributes defined during format generation can be modified by FHS.
  With a justified data transfer area, the attribute/length fields are aligned on a halfword boundary.

  Either all fields of +formats should be Unicode, or none of these fields should be able to contain Unicode characters. A mix of UNICODE and non-UNICODE fields is not recommended: If FHS detects that Unicode text is entered in a non-UNICODE field, it will issue the return code RC0424A4 that is global for the format but without being able to specify the concerned field. In these data structures, each character on a UNICODE field is represented by 2 bytes.

  Furthermore the application programmer is responsible for checking that the user only enters characters compatible with the EDF03 character set in a field used as UTM control field.

– with separate attribute blocks and field contents (#formats)

  This data transfer area consists of global attributes, field attributes and field contents. The field attributes can be controlled individually and are separate from the field contents.

  A mix of UNICODE and non-UNICODE fields is fully supported.
  When generating such a data transfer area with IFG for a Unicode format, the only difference to a non-Unicode data transfer area is that the field content part corresponding to the Unicode input or output fields is generated as an area containing two bytes for each character of the field. For further information on the generation of these data structures, please refer to the “IFG for FHS” manual. Today, only the addressing aids for COBOL are supported.

Further details on this data transfer area are given in the following section.

Refer to the „Unicode in BS2000/OSD” introduction for a survey of the Unicode support in the BS2000/OSD as well as basic information on Unicode.

4.1 Data transfer area with separate attribute blocks and field contents

The data transfer area with separate attribute blocks and field contents provides a convenient means of supporting formatted interactive operation with its global attributes and field attributes.

After each formatting operation (input and output), the contents of this data transfer area always corresponds to the current format displayed on the screen.

When a new format is output, all the attributes in the data transfer area can be set to their default values by the application program by deleting to binary zeros or to blanks. Only those attributes that are to differ from their default setting need explicitly to be supplied with values prior to the new output.

Before differential outputs, the application program needs to modify only those positions in the data transfer area that are also to be modified in the displayed format.

Using the FHS service function ’Initialization of the Data Transfer Area’ (see page 79), the attributes in the data transfer area can be supplied at any time with the default values from the format without modifying the field contents.

For formats utilizing this data transfer area (#formats), the application program can control individual field attributes (individual attribute updating) as the individual attributes are displayed independently of one another. The formats do not require separate transfer areas for input and output.

The separation of attributes and field contents allows simple further processing of the field contents without the need to mask out the field attributes.

There is no need for the application program to retain this data transfer area between output and input as during input formatting the complete data transfer area is always reconstructed from the restart area.

All the attribute values in the data transfer area (apart from pure numeric entries) have symbolic names. They are made available to the application program by FHS as an attribute value set as follows:

– for ASSEMBLER, as the macro MAVAL,
– for COBOL, as the copy element FHSAVAL.

For other programming languages, the attribute value sets must be created by the user.

Structure of the data transfer area with separate attribute blocks and field contents

This data transfer area is subdivided into:

– global attributes
– field attributes
– field contents

4.1.1 Global attributes

This part of the data transfer area contains those global attributes that are accessible to the application program. IFG addressing aids insure the necessary alignment (dependent on the language used). Global attributes that are incorrectly supplied with undefined values by the application program result in FHS error return codes (see page 513ff) .

Start parameters can be predefined for an application at the start of formatting. These define application-specific default values for the global attributes (differing from the FHS default values) and also other formatting attributes not present in the data transfer area. Start parameters are assigned as follows:

– for UTM applications by means of start parameters (openUTM as of V3.1 see page 81)
– for ASSEMBLER applications by means of certain MDCBL or MUCBL parameters
– for COBOL applications by means of certain FHS-INIT-PAR parameters

Missing start parameters are replaced by the FHS default values.

The functions and the permissible attribute values are described for all the global attributes defined in the following. The order in which they are arranged should be taken from the relevant addressing aid (see page 503ff).

There are further global attributes for FHS-DE, which can be used only in conjunction with openUTM Version 3.3 or later. These global attributes are described in the section “Global attributes of a DE format” on page 102.

The following applies to all global attributes: Deleting with NULL characters or blanks produces the default value.

4.1.1.1 Global attributes for the formatting acknowledgments

These global attributes are supplied during each formatting operation; their value prior to the formatting operation is irrelevant.

FHS RETURNCODES

The following three global attributes apply to the FHS return codes:

– MAIN RETURNCODE
– ERROR CATEGORY
– ERROR REASON

The values of these global attributes are always integers. The values and their significance are summarized in the table starting on page 513.

FORMATTING INDICATORS

FIELDS MODIFICATION indicator

Values:

NOT MODIFIED
No fields were modified; reset value for output formatting and for the service function ’Initialization of the Data Transfer Area’.

MODIFIED
At least one field was modified.

This global attribute always applies to the currently entered fields only.

FIELDS DETECTION indicator

Values:

NOT DETECTED
No fields were detected; reset value for output formatting and for the service function ’Initialization of the Data Transfer Area’.

DETECTED
At least one field was detected.

This global attribute always applies to the currently detected fields only.

FIELDS VALIDATION indicator

Values:

VALID
All fields in the format were checked by the edit routine and found error-free, and all fields with mandatory input were entered.

INVALID
At least one field of the format has not yet been checked by the edit routine or was found to be errored, or at least one field of the format with mandatory input has not yet been entered. Reset value for output formatting and for the service function ’Initialization of the Data Transfer Area’.

This global attribute always provides the global information for all fields in the data transfer area, including those fields that are not affected by the current formatting operation. The application program can recognize errors in individual fields by way of the field attribute ’Edit State’ and react accordingly.

USER EXITROUTINE RC

Here FHS provides the contents of the user exit interface return code byte after the formatting operation, if an exit routine is required for the formatting. The application program has sole responsibility for initializing the user exit interface return code byte before a formatting operation and for supplying values within the exit routine which must be in a form appropriate to the programming language used.

If no exit routine is required for a formatting operation, FHS supplies the reset value SPACES. This value is also supplied in the case of the service function ’Initialization of the Data Transfer Area’.

UNDEFINED VALUES

Values:

NOT UNDEFINED
No field contains an undefined value.

UNDEFINED
An undefined value was entered in at least one field.

This global attribute always applies only to the fields actually entered.

INPUT IDENTIFICATION

INPUT KEY CLASS

Values:

INPUT KEY	Normal input message
F-KEY	Function message
K-KEY	Short message
POS-RM	Positive printer acknowledgment
NEG-RM	Negative printer acknowledgment
INPUT NONE	Reset value for output formatting and for the service function ’Initialization of the Data Transfer Area’.

INPUT KEY NUMBER

The values depend on the values of the global attribute ’Input Key Class’ and are listed in the table below. The reset value for output formatting, and also for the service function ’Initialization of the Data Transfer Area’, is 0.

Input key class	Input key number	Cause of input
INPUT KEY	always 0	Key DÜ [1]
F-KEY	1 through 24	Key F1 through F24 [1]
K-KEY	1 through 24	Key K1 through K14 [1]
POS-RM	always 0	pos. printer acknowledgment
NEG-RM	always 0	neg. printer acknowledgment

[1] and all input functions mapped onto these keys

Note

Printer acknowledgement is not supported for RSO printers.

4.1.1.2 Global attributes for device initialization (DEVICE CONTROLS)

INIT CONTROL

Values:

NO INIT Output without initialization

FIRST INIT Initialization at start of output

LAST INIT Initialization at end of output

BOTH INIT Initialization at start and end of output

DEFAULT as ’FIRST INIT’

Initialization at the start of the output signifies a fresh screen display in the case of Data Display Terminals, fields having the field contents ’LOW VALUE’ are displayed with output fill characters, and signifies a form feed (with continuous forms) or a single-sheet feed in the case of printer terminals.

Initialization at the end of the output signifies ejection of the single sheet in the case of printer terminals where the printer supports this function. With printer terminals using continuous forms and with data display terminals this value is meaningless and is ignored by FHS.

The following table shows how the global attributes ’Initialization’ and ’Display Selection’ interact on printers.

Display Selection	**Init control**			
	NO INIT	**FIRST INIT**	**LAST INIT**	**BOTH INIT**
BOXB	no feed	page feed before printing	page feed after printing	page feed before and after printing
BOXL	no feed	page feed before printing	page feed after printing	page feed before and after printing
KEB	no feed	page feed before printing	no feed	page feed before printing
KEL	no feed	page feed before printing	no feed	page feed before printing

Note the following with regard to this global attribute:

– Differential outputs on Data Display Terminals require the value ’NO INIT’; any other value results in a new output.

– If during partial formatting an open output cycle condition occurs, this global attribute will be ignored.
– A new output always occurs on printer terminals; here the global attribute controls only the form feed.
– For single-sheet output, the global attribute ’Display Selection’ must be set correctly.
– For INIT CONTROL = NO INIT or LAST INIT, the parameter LEVEL SELECTION is irrelevant and will be ignored by FHS.
– For INIT CONTROL = NO INIT or LAST INIT, a 9022 printer retains the previous format.

INIT OPT

This global attribute is provided only for reasons of compatibility with future versions.

The following global attributes ’Tabulator Control’ and ’Function Lock’ are evaluated only for Data Display Terminals if the global attribute ’Init Control’ has the value ’FIRST INIT’ or ’BOTH INIT’. The values then apply until the next initialization at the start of the output. With partial formatting, the global attributes only at the start of the partial formatting cycle.

TABULATOR CONTROL

Values:

AUTOTAB
Automatic tabulator for the entire screen. On filling out the screen, the cursor automatically skips all the protected (non-detectable) fields.

NO AUTOTAB
No special tabulator function. The cursor must be positioned by the terminal user. This has no effect in formats for the 3270 that have been generated using “fast formatting”.

DEFAULT
As start parameter

Note for 3270

With formats generated using IFG for “fast formatting”, ’NO AUTOTAB’ is effective only for fields that are accessible to the program. Areas of the screen occupied by fields that are inaccessible to the program and also gaps between fields are always skipped by the cursor.

FUNCTION LOCK

Values:

DEFAULT	No functions locked
KEYLOCK	Keyboard input locked

The following global attributes ’VMI Control’ and ’HMI Control’ are evaluated only in the case of printer terminals. The table on page 536 shows the number of characters per line for the individual printers.

VMI CONTROL

Values:

VMI-1	Line spacing 1 (normal), 1/6 inch ≙ 4.23 mm
VMI-2	Line spacing 2 (close), 1/8 inch ≙ 3.17 mm
VMI-3	Line spacing 3 (minimum), 1/12 inch ≙ 2.12 mm
DEFAULT	As start parameter

HMI CONTROL

Values:

HMI-1	Character spacing 1 (normal), 1/10 inch ≙ 2.54 mm
HMI-2	Character spacing 2 (close), 1/12 inch ≙ 2.12 mm
HMI-3	Character spacing 3 (minimum), 1/15 or 1/17 inch ≙ 1.69 mm or 1.49 mm
DEFAULT	As start parameter

4.1.1.3 Global attributes for the output cycle (OUTPUT CONTROLS)

An output cycle comprises one or more output formatting operations for different partial formats and combines these to produce an output message. Output formatting for a full format is always regarded as a complete output cycle.

CYCLE CONTROL

Values:

DEFAULT
Continuation of an open output cycle or opening a new output cycle; reset value for output formatting.

CLOSE
Closing an open output cycle or opening and closing a new output cycle if only one partial format is to be output in the cycle.

The values of this global attribute are ignored in the case of full formats. Note however that the output formatting of full formats is permissible only when no output cycle is open at the same time. In the case of input formatting this global attribute has no significance.

COPY CONTROL

Values:

DEFAULT
No automatic hardcopy output; manual hardcopy (key LA1) is possible if the printer is connected locally to the data display terminal.

HARDCOPY GEN
Automatic hardcopy mode; the contents of the entire screen are output automatically to the hardcopy device following output of the message, provided the data display terminal has been generated with hardcopy (output to local or central hardcopy unit, depending on generation). The following should be set in PDN

– for UTM and DCAM: a suitable terminal characteristic with the appropriate OPCH parameter,
– for TIAM: either OPCH or TCHNG command (see “TIAM (TRANSDATA, BS2000)” User Guide).

HARDCOPY LOC
Automatic hardcopy mode; the contents of the entire screen are output automatically to a printer connected locally to the data display terminal following output of the message, regardless of how the data display terminal is generated.

This global attribute is always evaluated for full formats; with partial formatting it is evaluated only on opening the output cycle and then remains in force until the output cycle is closed.

ALARM CONTROL

Values:

DEFAULT
Output without alarm; resets any (visual) alarm that may remain.

ALARM
Output with alarm

This global attribute is effective only for devices that support an alarm function (visual and/or audible).

HOLE COLOR

Values:

WHITE
Spaces between fields are displayed with normal intensity.

GREY
Spaces between fields are displayed with reduced intensity.

UNDEFINED
Spaces between fields are output as protected fields which cannot be transferred. They are displayed with reduced intensity on Data Display Terminals.

DEFAULT
As specified in the start parameters.

This global attribute meaningful only for Data Display Terminals which support the function ’reduced intensity’ (9755, 9763) and for formats which were not generated for ’fast formatting’.

4.1.1.4 Formatting parameters (FORMATTING CONTROLS)

The following global attributes are evaluated during every formatting operation.

DISPLAY SELECTION

Values:

DEFAULT	Default value; with Data Display Terminals, the screen; with printer terminals, continuous forms
BOXB	Single-sheet cartridge, landscape
BOXL	Single-sheet cartridge, portrait
KEB	Form feed attachment, landscape
KEL	Form feed attachment, portrait

LEVEL SELECTION

Values:

LEVEL-1	
LEVEL-2	
LEVEL-3	
LEVEL-P	As ’DEFAULT’, but with integrated loading of the P keys.
DEFAULT	As ’LEVEL-1’

Integrated loading of P keys

Each time output formatting takes place for a data display terminal with P keys, the P keys can be loaded at the same time as the format is output regardless of whether a new or update output is concerned. To permit this, the global attribute ’Level Selection’ must be set to ’LEVEL-P’. FHS always resets this value to ’DEFAULT’ to prevent the P keys from being reloaded unnecessarily after every format output. In addition, the global attribute ’P-Key Set’ must be set to the P key format name. For terminals without P keys, the value ’LEVEL-P’ has the same effect as ’DEFAULT’. The P keys remain unaffected by a screen restart.

Note

UTM users are advised to use integrated loading of the P keys only in the sign-on conversation.

Evaluation of this global attribute depends on the global attribute ’Display Selection’. The interrelationship between these two global attributes is shown in the following table.

Display Selection	Level Selection		
	LEVEL-1, LEVEL-P, DEFAULT	LEVEL-2	LEVEL-3
DEFAULT	screen or continuous forms path 1	not permissible	not permissible
BOXB	single sheet cartridge 1 landscape	single sheet cartridge 2 landscape	single sheet cartridge 3 landscape
BOXL	single sheet cartridge 1 portrait	single sheet cartridge 2 portrait	single sheet cartridge 3 portrait
KEB	single sheet landscape	not permissible	not permissible
KEL	single sheet portrait	not permissible	not permissible

OUTPUT MODE

Values:

DEFAULT
Differential output of fields in the data transfer area in accordance with the field attributes for controlling the output. If the format is not yet displayed on the screen, new output occurs automatically.

RDIF
Reset with differential output for a format that is already displayed on the screen:

– ’RDIF’ output with no differences between the data transfer area and the restart area only results in the format being reset.

– All fields having no differences between the data transfer area and the restart area with the current field attributes ’DETECTABLE’ or ’UNPROTECTED’ (see field attribute ’Protection’) are reset; the field contents of all fields having the current field attribute ’UNPROTECTED’ are supplied with NULL characters (not with output fill characters) on the screen, in the data transfer area and in the restart area.

– All fields in which there are differences between the data transfer area and the restart area are output as for ’DEFAULT’.

– If the field attribute ’OUTPUT CONTROL’ has the value ’OUTPUT DATA’, the field contents are always output.

– ’RDIF’ has no effect upon the new output of a format.

FHS only has information as to which formats are currently displayed on the screen in the case of partial formatting. With full formats, FHS only performs a new output when initialization is required; otherwise a differential output always takes place.

Whenever #formats are output, all modified and marked fields on the screen are always reset to unmodified/unmarked, i.e. they are not read in again.

CURSOR CONTROL

Values:

DEFAULT
Default position for the cursor; the cursor is positioned on the first input location on the screen, regardless of any other information specified, i.e. to the first character in the first unprotected or detectable field on the screen, or to the first input location in the field for which the cursor was assigned during format generation. The global attribute ’Cursor Position’ is not evaluated.

FIELD CURSOR
Evaluation of the field attribute ’Cursor’. If the field attribute is not present in dynamic form, the cursor is positioned on the first input location on the screen. The global attribute ’Cursor Position’ is not evaluated.

EDIT CURSOR
The cursor is positioned on the first field of the format with an edit function error. If there is no field with an edit function error, or if initialization of the display is required at the same time, the cursor is positioned on the first input location on the screen. The global attribute ’Cursor Position’ is not evaluated.

REL CURSOR
Relative global cursor positioning; the global attribute ’Cursor Position’ contains the displacement to a data character in the field contents section to which the cursor is to be positioned on the screen. The displacement is relative to the start of the field contents section. As ’DEFAULT’ if the global attribute ’Cursor Position’ has an invalid value.
If the global attribute ’Cursor Position’ contains a displacement to an address that corresponds to either the first or the second byte of a Unicode character, the cursor will be positioned on this character on the screen.

If ’REL CURSOR’ with a valid cursor position has been used once in the output cycle, the most recent relative global cursor positioning in the output cycle always applies.

The field-specific cursor attributes are effective only in the case of currently unprotected fields and detectable fields.

In the case of global positioning of the cursor in fields that, due to the edit routine, have differing field lengths in the display and the addressing aid, the same displacement applies to the position of the cursor in the display as in the addressing aid. This means that with fields having a greater display length the cursor cannot be positioned on the last characters in the field, and that with fields having a shorter display length the user must insure that the cursor is not inadvertently positioned on the next field.

Global cursor positioning is always effective, regardless of other field attributes. It is the responsibility of the application program to check that the cursor is not positioned on protected fields.

CURSOR POSITION

This global attribute is evaluated only when the global attribute ’Cursor Control’ has the value ’REL CURSOR’.

USER EXIT CONTROL

Values:

NO UEXIT
User exit routine is not executed.

OUT UEXIT
User exit routine is executed during output formatting, if required.

IN UEXIT
User exit routine is executed during input formatting, if required.

BOTH UEXIT
User exit routine is executed during input and output formatting, if required.

DEFAULT
As start parameter, default value is ’NO UEXIT’.

This global attribute must be set during output formatting in all cases; it is not possible to change the value between output and input because in the case of the latter the corresponding global attribute in the restart area is applicable.

STARTLINE

In this global attribute you enter the vertical start position (line number) for a format. The format is output with this start position, thereby enabling a format to be output at various start positions on the screen. This means that a partial format can be present on the screen at several start positions simultaneously. When creating formats with variable start positions, you must observe the rules governing partial formatting plus the following:

– the format name may comprise a maximum of 7 characters,

– the format must be a #format and have a start line > 0 (specified in IFG), and

– application preparation may not take place for “fast formatting”.

Input formatting for partial formats with variable start positions can only take place sequentially. A partial format cannot be freely selected by means of format names.

The value for this global attribute is an integer. n is a line number for the format’s start line; 0 is the static default line of the format. A value must be selected for n which does not result in a format being shifted beyond the last line on the screen, otherwise FHS will issue a return code.

4.1.1.5 Other parameters

P-KEY-SET

In this global format you enter the name of the P key format. It is evaluated only if the global attribute ’Level Selection’ is assigned the value ’LEVEL-P’. If the global attribute only contains default characters (8 times C’ ’ or 8 times X’00’), the terminal’s P keys remain unchanged. All other values are interpreted by FHS to be a name specified for a P key format generated with PLUS.

The following requirements must be satisfied:

– The format application file currently assigned must contain a P key format generated with PLUS under the specified name.

– When this P key format is created, the control statement MSG may not be used.

– The P key format must be compatible with the terminal. If the format was created for “fast formatting”, the P key format must be compatible with the format’s device type.

If these requirements are not satisfied, FHS issues a return code.

4.1.1.6 Handling of global attributes

The following table illustrates how the individual global attributes are handled before and after formatting.

Note

The attribute values after output formatting and before input formatting are not accessible to the UTM user.

<table>
<tr><th rowspan="3">Global attribute</th><th colspan="4">Attribute value</th></tr>
<tr><th colspan="2">output formatting</th><th colspan="2">input formatting</th></tr>
<tr><th>before</th><th>after</th><th>before</th><th>after</th></tr>
<tr><td>FIELDS DETECTION
FIELDS MODIFICATION
INPUT KEY CLASS
INPUT KEY NUMBER</td><td rowspan="2">any value, not evaluated by FHS</td><td>FHS provides de reset value</td><td rowspan="4">any value, not evaluated by FHS</td><td rowspan="3">FHS provides current value of global attribute</td></tr>
<tr><td>FIELDS VALIDATION
ERROR CATEGORY
ERROR REASON
MAIN RETURNCODE
USER EXITROUTINE RC
UNDEFINED VALUES</td><td rowspan="2">FHS provides current value of global attribute</td></tr>
<tr><td>ALARM CONTROL
STARTLINE
OUTPUT MODE
DISPLAY SELECTION
USER EXIT CONTROL
FUNCTION LOCK
HOLE COLOR
INIT CONTROL
INITIALIZATION SCOPE
COPY CONTROL
CURSOR POSITION
P-KEY-SET
CURSOR CONTROL
TABULATOR CONTROL
HMI CONTROL
VMI CONTROL</td><td rowspan="3">set by application program; with undefined values, error return code</td></tr>
<tr><td>CYCLE CONTROL</td><td>FHS provides defined reset value</td><td>FHS provides defined reset value</td></tr>
<tr><td>LEVEL SELECTION</td><td>FHS provides current value of global attribute or defined reset value</td><td>FHS provides current value of global attribute or defined reset value</td></tr>
</table>

4.1.2 Field attributes

This part of the data transfer area contains the field attribute blocks for the fields in the format. A field attribute block contains all the field attributes for a field that are accessible to the application program, combined to produce field attribute groups. A field attribute block is assigned to each field in the format definition. When generating the format with IFG you specify which field attribute groups the field attribute block is to contain. The following applies to all field attributes: deletion with NULL characters or blanks produces the default value.

During format generation with IFG, selection of the field attribute groups should be given careful consideration. The application program is optimized in respect of memory requirement and run time provided that only those field attribute groups that are actually required are present in the addressing aid.

4.1.2.1 Field attribute group BASIC ATTRIBUTES

This field attribute group is always present with #formats. It comprises the following field attributes:

INPUT STATE

Values:

MODIFIED
Field modified by terminal user or field with automatic input and the field contains relevant characters.

CLEARED
As ’MODIFIED’, but no relevant characters in the field (only input fill characters).

DETECTED
Field detected by terminal user.

UNDEFINED
The field is filled with substitute characters for an undefined value or, if ’fast detection’ was selected for this field, the first position of the field contains the substitute character.

NOT TOUCHED
Field has been neither modified nor detected. This value is also the reset value for new output or when resetting all the fields in the format, and also for the service function ’Initialization of the Data Transfer Area’.

This field attribute specifies how the field has most recently been entered since the last new output of the format or since the format was last reset.

The application program can set its own processing identifiers in this field attribute, e.g. for its own additional checks. Valid processing identifiers are any characters apart from the uppercase letters A - Z. Note here that such processing identifiers are overwritten by one of the above values in the case of new output or resetting of the format, and also with the service function ’Initialization of the Data Transfer Area’ (all fields, or specific fields through the input of fields).

INPUT STATE ACT

Values:

MODIFIED
Field modified by terminal user or field with automatic input and the field contains relevant characters.

CLEARED
As ’MODIFIED’, but no relevant characters in the field (only input fill characters).

DETECTED
Field detected by terminal user.

UNDEFINED
The field is filled with substitute characters for an undefined value or, if ’fast detection’ was selected for this field, the first position of the field contains the substitute character.

NOT TOUCHED
Field has been neither modified nor detected. This value is also the reset value for any output, and also for the service function ’Initialization of the Data Transfer Area’.

This field attribute applies only to the currently entered fields. In contrast to the field attribute ’INPUT STATE’, the field attribute ’INPUT STATE ACT’ is set to the reset value ’NOT TOUCHED’ for any output (new output or differential output). Fields with automatic input are flagged as entered for any input.

This field attribute will only be evaluated by the application program when entered fields are processed immediately. If the application program does not process all entered fields until all fields are error-free, it is sufficient to evaluate the field attribute ’INPUT STATE’ in order to recognize all the entered fields.

With differential outputs, all fields having a value here other than ’NOT TOUCHED’ and having the value ’VALID’ in the field attribute ’Edit State’ are output again without checking for differences. This automatic re-output causes the edited display of all latterly entered valid field contents. Invalid field contents remain unchanged on the screen unless the application program forces a differential output on the basis of an error flag in the field contents.

If this field attribute is assigned the value 'NOT TOUCHED' by the application program before the differential output, the automatically edited re-output can be suppressed on a field-specific basis. Regardless of this, the field attribute differences are taken into consideration during differential output, i.e. modification of the attributes is always possible in the case of such fields.

EDIT STATE

Values:

VALID
Field contents checked and found error-free, or no edit routine required

INVALID
Field contents checked and found errored

MUST ERROR
Field with mandatory input not yet entered

NOT CHECKED
Field contents not checked; reset value for service function 'Initialization of the Data Transfer Area'.

Fields having no edit function are immediately set to the value 'VALID' in the case of new output or resetting of the format. This value is not altered by an input or differential output.

OUTPUT CONTROL

Values:

DEFAULT
Differential output (default value); FHS recognizes differences between the data transfer area and the restart area and decides internally which parts are to be output. One of the aims of this is to minimize the message length. New output of the format is treated as a difference case for all fields, i.e. everything is always output. After the service function 'Initialization of the Data Transfer Area' this value is set for all fields.

OUTPUT INIT
Initialization of the field contents, i.e. the field contents are supplied with the standard field contents from the format definition. These consist of either output fill characters or the text string for text fields the program can access. No output edit functions are performed.

OUTPUT DATA
Unconditional output of the field contents. Effective only if the global attribute 'Output Mode' is assigned the value 'RDIF', otherwise like 'DEFAULT'.

OUTPUT UNDEFINED
The field is output with the substitute character for the undefined value.

Any change in this attribute value results implicitly in the complete output of the field (all field attributes and the field contents). Exception: when the attribute value is changed from ’OUTPUT DATA’ to ’DEFAULT’ this is not interpreted as a change of attribute if the global attribute ’Output Mode’ is assigned the value ’RDIF’. When this occurs, the field attributes ’INPUT STATE’ and ’INPUT STATE ACT.’ are set to ’NOT TOUCHED’ for specific fields and the field attribute ’Edit State’ is set in accordance with the result of the check, where the value ’INVALID’ on output results in termination with a return code. The field attribute value is not changed by FHS.

If the attribute value is ’OUTPUT INIT’, the field contents in the data transfer area may contain data that is required for processing in the program but is not to be transferred or displayed. Different terminal users may thus have differing access authorization for the fields of a format, for example. Using ’OUTPUT INIT’ it is also possible to optically mask out individual fields of the format and make them reappear if blank is the output fill character and/or the field attributes ’Protection’ and ’Visibility’ have appropriate values. The application program can thus control the displayed information, depending on the dialog. It is thus possible, for example, to mask out fields in the format that are not required for the current dialog step.

4.1.2.2 Field attribute group FIELD INPUT

INPUT CONTROL

Values:

NORMAL IN
Normal input

MUST IN
Field with mandatory input; the field must be modified or detected by the terminal user. In the case of any output this value causes activation of the field as a mandatory input field. On activation, FHS sets the value to ’POTMUST IN’.

POTMUST IN
Field with potential mandatory input; FHS activates the field as a mandatory input field only in the case of new output or resetting of the format. Setting ’MUST IN’ to ’POTMUST IN’ on activation of mandatory input fields means that the one-off input of such a mandatory input field will suffice even in the event of multiple differential output dialog steps.

AUTORET IN
Automatic input; fields having this value are always reported as currently input in the field attribute ’Input State Act’ during input formatting. Unprotected fields with automatic input are output on the data display terminal as unprotected fields; however, a physical input is possible only in the event of modification or detection by the terminal user. Protected fields with automatic input are output as protected fields; no physical input is possible.

DEFAULT
Default value from the format

If the field attribute group ’Field Input’ is not present, mandatory input applies to any input operation affecting fields with mandatory input that were generated using IFG.

Mandatory input fields must always be unprotected. IFG does not permit any other combination during format generation. If mandatory input fields are protected by way of attribute modification, FHS returns an attribute error.

The attribute ’Mandatory Input’ can optionally be assigned for a field during format generation (static mandatory input) and can be modified dynamically by means of the field attribute ’Input Control’. Mandatory input fields are activated upon output by FHS when:

– the field attribute ’Input Control’ has the value ’MUST IN’, or ’DEFAULT’ in the case of static mandatory input;
– the field attribute group ’Field Input’ is not present and the field is a field with static mandatory input;
– the field attribute ’Input Control’ has the value ’POTMUST IN’, but only in the case of new output or resetting of the format.

In the case of a differential output, active fields with mandatory input remain active; fields with potential mandatory input are not reactivated unless the application program requires this.

PROTECTION

Values:

UNPROTECTED
Unprotected; the field contents can be modified via the keyboard. Input of all characters is permitted. In the case of fields having the static attribute ’NUM Lock’, only the input of numeric characters (numeric in the sense of the data display terminal) is permitted.

PROTECTED
Protected field; the field contents cannot be modified via the keyboard.

ASKIP
Significant only for the 3270 display terminal; as ’PROTECTED’, except that the cursor skips the field.

DETECTABLE
Protected selection field

DEFAULT
In output formatting the value ’DEFAULT’ is replaced by the default value from the format; in input formatting this value has no significance.

4.1.2.3 Field attribute group DISPLAY CONTROL

INTENSITY

Values:

HIGH INTENSITY
High intensity; with printer terminals this value results in bold print, providing the device supports this function.

NORMAL INTENSITY
Normal intensity

DEFAULT
Default value from the format; can be changed by means of a local device setting.

On Data Display Terminals having color monitors (9752, 9754, 9763), 'HIGH INTENSITY' is displayed in green (red) and 'NORMAL INTENSITY' in yellow (white). The colors in parentheses apply when, in addition, the field attribute 'underline' has the value 'UNDERLINED'. The choice of color can be changed by means of a local device setting.

On the 9763 Data Display Terminal with color monitor, this applies only to monochrome formats; with color formats, only the field attribute 'colour' applies.

VISIBILITY

Values:

VISIBLE
Field contents are fully visible.

SIGNALING
Field contents are displayed flashing; with printer terminals this value results in shaded printing, providing the device supports this function.

INVISIBLE
Field contents are invisible, and non-printable in respect of hardcopy.

DEFAULT
Default value from the format

If dynamic attribute assignment results in the invalid combination of 'SIGNALING' and 'Selection Field' for a particular field, a field attribute error is reported.

UNDERLINE

Values:

DEFAULT
Default value from the format

NOT UNDERLINED
Field contents are not displayed underlined.

UNDERLINED
Field contents are displayed underlined. With some terminals, an alternative display mode is selected (see table).

The following table indicates how the field contents are displayed on the various Data Display Terminals when the field attribute 'underline' has the value 'UNDERLINED'.

Terminal	Field contents display mode
8160	italics
9750	underlined
9752	white / red 1)
9754	white / red 1)
9755	underlined
9763	underlined
3270	-
9001	underlined
9002	italics
9003	italics
9004	underlined
9011	underlined
9012	underlined
9013	underlined
9022	underlined
PCL	underlined
3287	underlined

1) dependent on field attribute 'intensity'

The choice of color can be changed by means of a local device setting.

INVERSE

Values:

DEFAULT
Default value from the format

NOT INVERSE
Field contents are not displayed in inverse video.

INVERSE
Field contents are displayed in inverse video.

This field attribute is of significance only for the 9763 Data Display Terminal; with all other terminals this value is ignored by FHS.

4.1.2.4 Field attribute group CURSOR

CURSOR

Values:

CURSOR
The cursor is positioned on the first input position in the field; with output formatting, this value is reset to 'DEFAULT' after evaluation.

HOLD CURSOR
The cursor is positioned on the first input position in the field; with output formatting, this value is not reset after evaluation.

NO CURSOR
No cursor in the field

DEFAULT
As 'NO CURSOR'.

This field attribute is evaluated only for Data Display Terminals and then only if field-specific positioning of the cursor is required by the presence of the value 'FIELD CURSOR' in the global attribute 'cursor control'.

If the cursor is required for more than one field in a format, the first specification is applicable in the order in which the fields are processed.

For programming languages that support address computation, positioning of the cursor by means of the global attributes 'cursor control' and 'cursor position' is recommended as the field attribute 'cursor' can then be omitted and the transfer area is shorter.

4.1.2.5 Field attribute group FIELD LENGTH

Field length in the data transfer area

For input formatting, or if the service function is used, FHS returns the length of the field, as specified in the field definition, in the data transfer area. For applications where the programming language allows access to the implicit length of a data field, this field attribute is not required. The data transfer area is thus shorter.

The field attribute group FIELD LENGTH is equal to the number of characters or half of the number of bytes in case of a UNICODE field, as specified in the field definition.

4.1.2.6 Field attribute group 'Attribute Combination'

This field attribute group is provided principally for existing application programs with +formats to enable them to utilize the functions of this data transfer area without the need for extensive conversion operations. New application programs should not use this field attribute group.

Note

This field attribute group is **incompatible** with the field attribute groups 'Field Input', 'Display Control' and 'Cursor'; IFG does not permit such combinations.

Attribute Combination

Update with MATUP for ASSEMBLER, or CALL "FHSATTR" or FHS-ATTRIBUTE-MOVE for COBOL. Bit attributes in the case of +formats for openUTM.

Individual attribute update is not possible; FHS regards all field attribute values as a valid attribute combination with the exception of the value binary zero which requires the standard attributes from the format definition. The field attribute 'Inverse' is not supported here.

Note the following when using this field attribute group:

- If a field has been declared a mandatory input field during generation with IFG (static mandatory field) and conflicts arise as a result of attribute modifications (e.g. protected, non-detectable mandatory field), FHS reports an attribute error.
- If the value 'NUM' (numeric field) is entered for an alphabetic field by way of attribute modification, no valid input is possible for this field.
- When using this field attribute group, the field attributes of all fields are always output in the data transfer area.

4.1.2.7 Field attribute group COLOUR

COLOUR

Values:

DEFAULT	Default value from the format
RED	
GREEN	
YELLOW	
BLUE	
MAGENTA	
CYAN	
WHITE	
NO COLOUR	No color, i.e. normal display mode for data display terminal

The colors correspond to the color assignment applicable at the time of delivery of the data display terminal. The color assignment can be changed by means of a local device setting.

FHS ignores this field attribute group in the case of devices that have no color display capability.

4.1.2.8 Field attribute group 'Edit return code'

EDIT-RC

If errors are detected in the field contents (data errors) during field processing by edit routines, this field attribute supplies a field-specific edit return code. FHS initializes the value (C'00'). Other values may only occur if edit errors are detected for the field by edit routines. This field attribute is only valid when the associated field contents have been checked, i.e. the value of the field attribute 'Edit State' must be something other than 'NOT CHECKED'.

Note

Application programs not wishing to react in detailed fashion to edit errors do not require this field attribute; this results in a shorter data transfer area. The information contained in the field attribute 'Edit State' normally suffices.

Return codes

The return code consists of two printable characters and provides information on the course of the EDIT call.

The first character indicates a general classification while the second provides more detailed information on the cause of the error.

C’00’ EDIT execution error-free

C’10’ Invalid characters in field

C’19’ The field on the data display terminal contains characters that are not compatible with the base character set of the format
It is issued when a format containing at least one UNICODE field is output on the data display terminal (therefore in “Unicode mode”) and a Unicode string is entered in a field not foreseen for such a string.
See section “Which terminals does FHS work with?” on page 18

C’20’ Insufficient relevant characters entered

C’30’ Date error

C’31’ Limit error

C’32’ Day error

C’33’ Month error

C’34’ Year error

C’35’ Separator incorrect/missing

C’36’ Length error

C’37’ Hour error

C’38’ Minute error

C’39’ Second error

C’40’ Too many leading positions entered

C’50’ Too many decimal places entered

C’60’ Sign not permitted

C’61’ More than one sign in field

C’62’ Sign incorrect on output

C’63’ Only sign in field

C’70’ Digit separator not permitted

C’71’ Digit grouping error

C’80’ Decimal separator not permitted

C’90’ Error in arithmetic field

C’91’ Length error

4.1.2.9 Handling of field attributes

The following table illustrates how the individual field attributes are handled before and after formatting.

<table>
<tr><th rowspan="2">Field attribute</th><th colspan="4">Attribute value</th></tr>
<tr><th colspan="2">output formatting</th><th colspan="2">input formatting</th></tr>
<tr><th></th><th>before</th><th>after</th><th>before</th><th>after</th></tr>
<tr><td>INPUT STATE</td><td rowspan="3">Application program can insert permissible values before output formatting and thus control output formatting</td><td>Reset value for new output or resetting of a for mat, other wise unchanged</td><td rowspan="11">Any value, not evaluated by FHS</td><td>FHS supplies current value if the field is affected, otherwise unchanged</td></tr>
<tr><td>INPUT STATE ACT.</td><td>Reset value for any output formatting</td><td>FHS supplies current value if the field is affected, otherwise reset value</td></tr>
<tr><td>EDIT STATE</td><td rowspan="10">FHS supplies current value</td><td rowspan="10">FHS supplies current value</td></tr>
<tr><td>OUTPUT CONTROL</td><td rowspan="8">Supplied by appli-cation program, error return code in case of undefined values</td></tr>
<tr><td>INPUT CONTROL</td></tr>
<tr><td>PROTECTION</td></tr>
<tr><td>INTENSITY</td></tr>
<tr><td>VISIBILITY</td></tr>
<tr><td>UNDERLINE</td></tr>
<tr><td>INVERSE</td></tr>
<tr><td>CURSOR</td></tr>
<tr><td>Field Length in the Data Transfer Area</td><td>Any value, not evaluated by FHS</td></tr>
<tr><td>Attribute Combination</td><td rowspan="2">Supplied by appli-cation program, error return code in case of undefined values</td></tr>
<tr><td>COLOUR</td></tr>
<tr><td>EDIT-RC</td><td>Any value, not evaluated by FHS</td><td>New value for field with edit function, otherwise unchanged value</td><td>Any value, not evaluated by FHS</td><td>New value for field with edit function, otherwise unchanged value</td></tr>
</table>

Note

The attribute values after output formatting and before input formatting are inaccessible to the UTM user.

4.1.3 Field contents

This part of the data transfer area contains the contents of the fields of a format. Before each output formatting operation (new output or differential output) the application program can supply the field contents in the data transfer area with any desired data.

- Handling the field contents without calling an exit routine

  The field contents are not changed in the data transfer area by output formatting. The field contents in the restart area are identical. All editing functions affect the display only.

  During each input formatting operation, firstly all the field contents in the data transfer area are overwritten by the corresponding field contents from the restart area. Then the field contents of all the entered fields are aligned in the data transfer area, supplied with input fill characters and, if required, returned in edited form. The field contents in the restart area are identical.

- Handling the field contents by calling an exit routine

  In the case of output formatting of fields using an exit routine, the exit routine affects only the display of the field contents. In the data transfer area and in the restart area the field contents appear as in the case of fields where no exit routine is used.

  With input formatting the field contents appear as handled by the exit routine, in the data transfer area and in the restart area.

  Input data for the exit routine are the completely edited field contents.

- Handling of field contents using edit functions

  An edit error during output formatting results in aborted formatting. FHS supplies a return code. This type of error (errored data in the application program) should be excluded by testing the application program.

  The field contents of incorrectly entered fields are overwritten with NULL characters during input formatting. This means that errored data does not reach the application program. If formats containing such fields are re-output, or in the event of a restart, output fill characters appear on the screen.

Text fields that are accessible to the program are treated as normal fields accessible to the program in the case of a differential output. Where a new output takes place or the format is reset, the fixed text is always output.

An exit routine and edit functions are not mutually exclusive. When an exit routine is used, the responsibility lies with the application program. But the exit routine can no longer be called for fields in which edit errors have been detected. The edit functions are executed before the exit routine in the case of both inputs and outputs.

4.1.4 Validity error dialog for input errors

Evaluation of the field attributes ’Input State Act.’ and ’Edit State’ is prerequisite for a validity error dialog. Fields containing edit errors are retained as most recently entered via the keyboard; valid entered fields are output in edited form. The global attribute ’Fields Validation’ globally reports edit errors; the global attribute ’Cursor Control’ provides the means for positioning the cursor to the first errored field. The application program can search the data transfer area for fields containing edit errors and flag these fields by modifying a field attribute and/or the field contents. This flag will cause the field to be output when a differential output takes place.

In many cases there is no need for field-specific flagging as the edited display of valid entered fields suffices as a flag. After these actions have taken place the error dialog message is created by an output formatting operation (value ’DEFAULT’ in the global attribute ’Output Mode’ and ’NO INIT’ in the global attribute ’Init Control’).

Normally the application program will not continue processing the input data from a format until all the fields are error-free (value ’VALID’ in the global attribute ’Fields Validation’). Here, the field attribute ’Input State’ flags all fields that have been input at least once since new output or resetting of the format. However, the application program can also process each input immediately. Here, the field attribute ’Input State Act.’ flags all currently entered fields. In the case of partial formats the error message is created by means of a differential output cycle that contains all, or at least all the errored, partial formats displayed on the screen.

Example of an error dialog

If during the input formatting for a format the global attribute ’Fields Validation’ is reported as being other than ’VALID’, at least one field containing an input error is present on the screen.

The application program then sets

– the global attribute ’Init Control’ to ’NO INIT’ (differential output)
– the global attribute ’Output Mode’ to ’DEFAULT’
– and the global attribute ’Cursor Control’ to ’EDIT CURSOR’ (cursor is positioned in the first errored field).

The data transfer area can then be output on the screen without any other changes by output formatting. All currently valid entered fields are displayed in edited form, all errored fields remain unchanged, and the cursor is positioned in the first errored field.

This process should be repeated in a loop until the global attribute ’Fields Validation’ has the value ’VALID’.

The application program is of course able to make any desired modifications to the field attributes and field contents in the data transfer area before any output; all differences are then additionally output. Note here however that the output of new field contents to an errored field eliminates the error (except in the case of mandatory input fields where no input was made).

It is possible to deal with partial formats in exactly the same manner, with the proviso that as a minimum requirement all errored partial formats be output on the screen through output formatting. It is however advisable to output all the partial formats on the screen as only in this way is it possible to insure that all currently valid entered fields are output on the screen in edited form. The cursor is positioned in the first errored field that is found in the output cycle. During the error dialog it is thus advisable to output the partial formats on the screen according to their sequence.

4.1.5 Partial formats

For partial formats using the data transfer area with separate attribute blocks and field contents a restart area of adequate size must be available as such formats are always formatted by FHS with the restart function; see also the global attribute ’Cycle Control’.

Input formatting of partial formats with variable starting positions is only possible sequentially for the partial formats, that means in the order they appear on the screen. It is not possible to select partial formats randomly with the aid of their format names.

It is possible in principle for screens to be composed of a mixture of partial formats with and without separate attribute blocks and field contents. In order to avoid problems in such cases each output cycle must begin with a partial format that uses the data transfer area with separate attribute blocks and field contents.

Partial formats that use the data transfer area without separate attribute blocks and field contents should be formatted by means of the appropriate MDCBL parameters for ASSEMBLER (see page 232) or the FHS-MAIN-PAR parameters for COBOL (see page 303). For UTM applications, see the MPUT call in the openUTM manual “Programming Applications with KDCS for COBOL, C and C++ ”.

If these rules are not observed, control over screen-global functions (e.g. ATAB, KEYLOCK) is not always ensured, or FHS outputs a return code. Using a mixture of partial formats that employ differently structured data transfer areas is therefore not recommended.

4.1.6 Initialization of the data transfer area

The data transfer area with separate attribute blocks and field contents can be reset to the initial status at any time by means of the service function ’Initialization of the Data Transfer Area’, i.e. all field attributes are set in accordance with their default values in the format.

The field contents remain unchanged, as do the global attributes with the exception of the global attributes for the formatting acknowledgments. It is thus also possible to reset to the initial status data transfer areas that have already been supplied with data. New output is recommended for the output formatting that follows an initialization of the data transfer area (’FIRST INIT’ in the global attribute ’Init Control’).

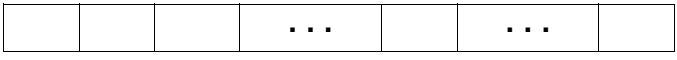
4.2 Other data transfer areas

The other data transfer areas are distinguished according to those that have no attribute fields (*formats) and those that have attribute fields (+formats).

Data transfer area for *formats

The data transfer area for *formats contains the variable fields (input/output fields) of the format. It may however also contain text fields if 'Yes' is explicitly entered in IFG for "Field Accessible to Program".

The data transfer area has the following simple structure



where one box is assigned to one field (see also the structure of the addressing aid).

Data transfer area for +formats

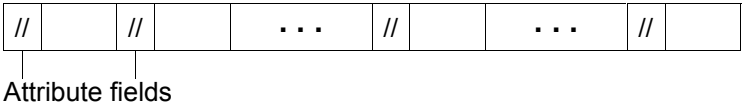
This data transfer area has an additional feature when compared to the data transfer area for *formats in that a 2-byte attribute field is placed before each field.

The attribute field can be changed by the application program before output formatting, e.g. in order to set a field to flashing or bright.

Use of this attribute field is described in the reference manuals for the programming languages.

During input formatting, if desired, FHS supplies the effective input length of the field or the defined field length, dependent on the operand EFFLEN in the MDCBL macro, or MAP-EFF-LEN in FHS-MAIN-PAR.

The data transfer area then has the following structure:



It is also possible to request "aligned" in IFG for +formats. In this case the data transfer area starts on a halfword boundary and each individual attribute field is also aligned on a halfword boundary.

The data transfer area has the same structure as above, except that a fill byte is inserted after fields having an odd byte length.

5 FHS for openUTM users

This chapter describes all the options available to you when your UTM application is working with FHS. You will learn:

- how the use of formats affects UTM program units
- how to generate mask systems using the FHS dialog extension (FHS-DE) which conform to the 'Alpha Style Guide', how these mask systems are programmed, and how they may be used by the terminal user
- how service functions, such as KDCFHS and KDCSCUR, are used
- how to link and load a UTM FHS application
- which FHS-specific start parameters can be specified at the start of a UTM FHS application
- which messages are issued by the FHSCON link module and their meaning.

5.1 How to use formats with openUTM

This chapter tells you

- which format types are available in openUTM
- how to use these formats with openUTM
- how to change the format attributes in the program
- how to format the output using screen functions and attributes
- how to enable format output in different languages

Format types

UTM enables you to use all types of data transfer areas which FHS can work with; see page 45. This means that you can use *formats, +formats, and #formats in your application. The characteristics of +formats and *formats are described on page 80. Further information on #formats is given starting on page 46.

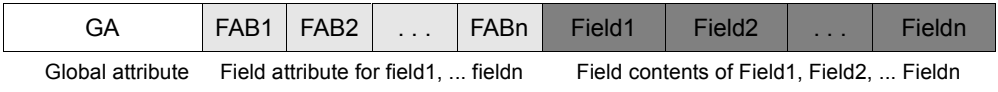
For information on using FHS in the UTM user exit FORMAT, refer to page 93.

Format identifier

When you want to use formats with UTM, you must supply the KCMF field of the KDCS parameter area with a format identifier; see section “Format types” on page 23. The format identifier consists of the prefix ’#’, ’+’, or ’*’ and the format name, e.g. *FORM1 for a *format, +Fig15 for a +format, or #screen4 for a #format. The format name should be a maximum of seven characters.

5.1.1 How to use #formats

Addressing aids are provided for formats created with IFG, which can be copied into the program. The message area has the following structure:



The possible global attributes are described starting on page 47, and the possible field attributes starting on page 62.

There are special #formats for formats (=masks) that conform to the Alpha Style Guide. For further details on how to generate and use masks that conform to the Alpha Style Guide, see section “FHS dialog extension” on page 99.

The important start parameters for #formats are described in section “Start parameters” on page 196.

Special points when using openUTM

K keys

If K keys are entered, this is not reported in the global attributes ’Input Key Class’ and ’Input Key Number’ under openUTM. When the first MGET call is issued, either the return code generated for the K key just entered (KDCDEF control statement SFUNC) or return code “19Z” for K keys that have not been generated is entered in the KCRCCC field. The K keys have the same meaning as described in the openUTM manual “Generating Applications” under the KDCDEF control statement ’SFUNC’.

F keys

When the first MGET call is issued, either the return code generated for the F key just entered (KDCDEF control statement SFUNC) or return code “19Z” for F keys that have not been generated is entered in the KCRCCC field. The current F key is not contained in the data transfer area in the global attributes ’Init Control’ and ’Input Key Number’ at this point. The entire data transfer area is the same as when the last output took place. Only with the second MGET call are the current F key and the format entry entered in the data transfer area. The F keys have the same meaning as described in the openUTM manual “Generating Applications”.

Global attributes

The screen functions KCREPL, KCREPR, KCALARM, and KCERAS are implemented in #formats by the following global attributes:

– Init Control (for the KCREPL function)
– Copy Control (for the KCREPR function)
– Alarm Control (for the KCALARM function)
– Output Mode (for the KCERAS function)

The global attribute ’Cycle Control’ has no effect in UTM program units.

MGET

#formats being displayed on the screen are always read with MGET even if no field has been modified or selected. The reason for this is that certain global and field attributes (return fields) always have to be updated. The input length (KCRLM) is always the full length of the data transfer area.

MPUT

If a #format is output with a length (KCLM) that is less than the length of the global attributes + field attributes, the conversation aborts with KCRCCC= “70Z” and a secondary FHS return code.

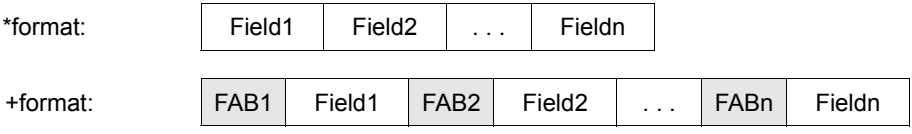
The KCDF field (screen output function) must always be supplied with a binary zero.

Fields with automatic entry

In #formats fields with automatic entry (’AUTORET IN’ in field attribute ’Input Control’) are only physically transferred if the field was modified on the terminal. With *formats and +formats, on the other hand, fields with automatic entry are always transferred. Thus in #formats the transaction code should always be entered in UTM control fields (specified in IFG) since openUTM is always supplied with these when automatic entry is used even if the terminal user has not modified them.

5.1.2 How to use *formats and +formats

Addressing aids are provided for formats created with IFG, which can be copied into the program. The message area has one of the following structures, depending on the format identifier:



FAB1, FAB2, ... Field attribute for field1, field2, ...
Field1, Field2, ... Content of field1, field2, ...

With “*” formats, you cannot change the field attributes using a program, whereas with “+” formats you can change the field attributes in the program; see page 86.

With output formatting the choice of FHS start parameters determines how fields are transferred, overwritten, or deleted, see section “Start parameters” on page 196.

5.1.3 Screen output functions for * formats and +formats

openUTM enables you to request specific screen functions together with a message output (only with +formats and *formats in formatted mode). You do this by setting the KCDF field of the KDCS parameter area to one of the values predefined for the relevant programming language. These are contained in the corresponding COPY or include elements (in COBOL: the KCDFC COPY element).

You can use the following functions in formatted mode:

Name	Function
KCREPL	REPLACE: The screen is erased prior to output and then rewritten.
KCERAS	Only variable fields should be deleted.
KCALARM	ALARM: An audible alarm is given when outputting. Only applies if the terminal has been configured accordingly.
KCREPR	Hardcopy output on local printer (LA1 function).
KCRESTRT	Screen restart after PEND RS.

If you use message segments, the entry is only valid for the first message segment; for the subsequent message segments you must enter 0 in the KCDF field.

The effect of KCERAS and KCREPL depends on the FHS start parameters selected, see page 196ff.

You can combine screen output functions by simply adding them, for example using the statement

```
KCDF = KCREPL + KCREPR + KCALARM
```

as the function values are defined numerically.

5.1.4 Modifying KDCS attributes with +formats

If you use “+” formats, you can change the field attributes of your formats by setting the desired value in the attribute field. This enables you, for example, to influence the display attributes of the format fields, and to send an incorrect input back to the data display terminal in flashing mode.

Important

1. If you use the same transfer area for input and output formatting, then you must set all the attribute fields you do not want to change to binary zero.
2. Always enter all attributes, since (unlike “#” formats) the individual attributes are not additive.

Evaluation of the attribute fields can be controlled via the FHS start parameters, see page 196.

All the following attribute modifications are contained in a data structure which can be copied into the program; two forms are provided for this purpose:

– **KC**name: default attribute combination

– **KC**abcd: other attribute combinations. These are combined from the initial letters of the possible values for “name”.

The following table shows which values you can set for “name” and “abcd”, and the corresponding attributes as per KDCS standard.

KDCS attribute combinations of the KCname form

KCname	Field attribute	Processing attribute	Display attribute	Additional attribute
KCALPH	all characters[1]	unprotected	bright	
KCNUME	numeric	unprotected	bright	
KCPROT	all characters	protected	normal	
KCUNPR	all characters	unprotected	bright	
KCNINT	all characters	unprotected	normal	
KCDINT	all characters	unprotected	not displayed	
KCHINT	all characters	unprotected	bright	
KCITAL	all characters	unprotected	undersc., bright	
KCSIGN	all characters	unprotected	flashing, bright	
KCDETE	all characters	protected	bright	markable
KCPREM	all characters	unprotected	bright	automatic return

[1] all characters = all printable characters

KDCS attribute combinations of the KCabcd form

KCabcd	Field attribute	Processing attribute	Display attribute	Additional attribute
KCAUH	all characters[1]	unprotected		
KCAPH	all characters	protected		
KCNUH	numeric	unprotected	bright	
KCNPH	numeric	protected		
KCAUN	all characters	unprotected		
KCAPN	all characters	protected		
KCNUN	numeric	unprotected	normal	
KCNPN	numeric	protected		
KCAUD	all characters	unprotected		
KCAPD	all characters	protected		
KCNUD	numeric	unprotected	not displayed	
KCNPD	numeric	protected		
KCAUI	all characters	unprotected		
KCAPI	all characters	protected		
KCNUI	numeric	unprotected	undersc., bright	
KCNPI	numeric	protected		

KCabcd	Field attribute	Processing attribute	Display attribute	Additional attribute
KCAUS	all characters	unprotected	flashing, bright	
KCAPS	all characters	protected	flashing, normal	
KCNUS	numeric	unprotected	flashing, bright	
KCNPS	numeric	protected	flashing, normal	
KCAUHD	all characters	unprotected		
KCAPHD	all characters	protected		
KCNUHD	numeric	unprotected	bright	markable
KCNPHD	numeric	protected		
KCAUND	all characters	unprotected		
KCAPND	all characters	protected		
KCNUND	numeric	unprotected	normal	markable
KCNPND	numeric	protected		
KCAUID	all characters	unprotected	undersc., bright	
KCAPID	all characters	protected	undersc., normal	
KCNUID	numeric	unprotected	undersc., bright	markable
KCNPID	numeric	protected	undersc., normal	
KCAUSD	all characters	unprotected	flashing, bright	
KCAPSD	all characters	protected	flashing, normal	
KCNUSD	numeric	unprotected	flashing, bright	markable
KCNPSD	numeric	protected	flashing, normal	
KCAUHP	all characters	unprotected		
KCAPHP	all characters	protected		automatic
KCNUHP	numeric	unprotected	bright	return
KCNPHP	numeric	protected		
KCAUNP	all characters	unprotected		
KCAPNP	all characters	protected		automatic
KCNUNP	numeric	unprotected	normal	return
KCNPNP	numeric	protected		

[1] “all characters” means: all printable characters

5.1.5 Partial formats

A screen display can be made up of two or more partial formats. The partial formats normally occupy only part of the screen. You have to issue a separate MPUT NT call for each partial format. If such a screen is to be read, then again you need a separate MGET call for each partial format.

The attribute “partial format” is defined when creating formats with IFG; addressing aids can be created in the same way as with normal formats.

5.1.6 Outputting partial formats with MPUT

If the screen is made up of two or more partial formats, you have to specify the value KCREPL in the KCDF field with the first MPUT NT (except for “#” formats, for which binary zero must always be entered). KCDF has to be set to binary zero for all subsequent MPUT calls, otherwise UTM aborts the conversation with KCRCCC = 70Z and KCRCDC = K606.

```
INIT
MPUT NT, KCMF = PARFOR1, KCDF = KCREPL + KCREPR
MPUT NT, KCMF = PARFOR2, KCDF = 0
MPUT NT, KCMF = PARFOR3, KCDF = 0
```

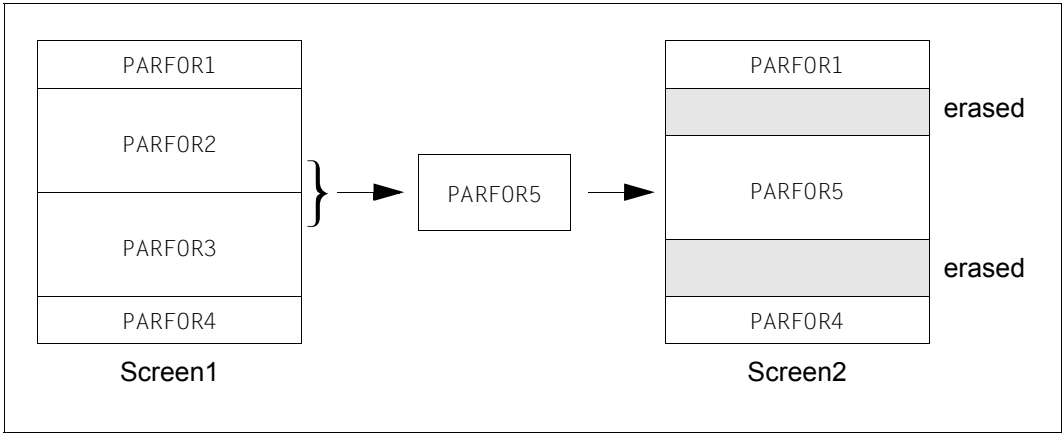
You are not allowed to mix formatted mode and line mode within a single message, otherwise UTM responds with KCRCCC = 75Z and aborts the conversation.

You are allowed to mix “*” formats and “+” formats. If a single format is specified for the first message segment, all subsequent message segments are ignored. If message segments with partial formats are followed by a message segment with a single format, UTM will report a formatting error. It is not recommended to mix * or +partial formats with #formats.

Updating a screen

You update a screen with one or more MPUT NT calls. With the first call, you may enter any value in the KCDF field except KCREPL, because KCREPL erases the entire screen (except for “#” formats, for which binary zero must always be entered). For subsequent MPUT NT calls, KCDF must be set to binary zero as if you were setting up a new screen.

If the format changes, then those old partial formats which are overlapped by the new partial formats are completely erased, i.e. even those parts of the old partial formats that are not overwritten.



You will find more details in section “Partial formats” on page 38.

With partial formats, it is not possible to switch between “*” and “+” formats which have the same name without KCREPL, otherwise this results in a formatting error.

The screen output function KCERAS can also erase unprotected fields in a partial format at output (see section “Screen output functions for * formats and +formats” on page 85).

Output of partial formats with FPUT/DPUT

Since, when a format name is specified, openUTM does not know whether a partial format is involved, it is possible to use FPUT/DPUT to output individual formats which were generated as partial formats. However, it is not possible to set up a screen from two or more FPUT/DPUT NTs, or by outputting with MPUT and updating with FPUT or DPUT.

The screen is cleared prior to every asynchronous output. openUTM responds to subsequent entries from asynchronous formats - except for UTM commands - by automatically restarting the screen.

5.1.7 Input formatting with partial formats

If a screen is composed of several partial formats, it can happen that the terminal user does not make an entry in one or more partial formats. In such a case, to simplify processing in the program UTM supplies in KCRMF of INIT, the name of the uppermost +, or *partial format with existing input. If no data is entered on the screen, UTM supplies the name of the uppermost partial format.

#Formats are always loaded if no data is available. Following the MGET or FGET call, KCRMF contains the name of the next partial format with input data. If there are no further partial formats with data, KCRMF contains the name of the last partial format with data, i.e. KCMF and KCRMF are identical.

If, however, you use “#” formats and movable partial formats, then reading must be performed until KCRCCC = 10Z is set, because a partial format can be used several times on the screen.

If the INPUT exit does not define otherwise, UTM removes the first 8 characters of the first partial format (with “*” formats) or the first 10 (with “+” formats) for the transaction code at the start of the conversation. Nothing is removed from subsequent partial formats or “#” formats.

Example

MPUT output with 3 partial formats PARFOR1, PARFOR2, PARFOR3;
data input in PARFOR1, PARFOR2

KDCS call:	**openUTM return information:**
INIT	
	KCRMF = PARF1
MGET KCMF = PARF1 KCLA = ...	
	KCRMF = PARF2 KCRCCC = 000 KCRLM = ... Data in message area
MGET KCMF = PARF3 KCLA = ...	
	KCRMF = PARF2 KCRCCC = 03Z KCRLM = 0
MGET KCMF = PARF2	
	KCRMF = PARF2 KCRCCC = 000 KCRLM = ... Data in message area
MGET ...	
	KCRCCC = 10Z

The way in which the fields are transferred depends on the FHS start parameters selected, see page 196.

It is permissible to enter a message to an asynchronous program from a screen with partial formats at the end of a dialog conversation (i.e. with PEND FI):

The input data may come from two or more partial formats. The asynchronous program must be written the same as the dialog program: the INIT supplies the name of the first partial format with input data in KCRMF. The program can then fetch the input data from one partial format at a time using separate FGET calls.

A dummy input message will cause the very first FGET to send 10Z to the program unit.

Input formatting of partial formats with variable start positions

As a format identifier can appear more than once when partial formats with variable start positions are used, the end can only be detected by means of return code KCRCCC="10Z".

5.1.8 FHS in the FORMAT user exit

The FORMAT user exit must always generate its own FHS macros MGMAP and MDCBL (see pages 217 and 232). In doing so the program must not execute the code generated by MGMAP. The CSTM operand must not be specified in the MGMAP macro; the IOLEN operand must be compatible with the TRMSGLTH and NB operands in the KDCDEF control statement MAX (see the openUTM manual “Generating Applications”).

The UTM control field cannot be used with any format type.

The DEVAR operand must be specified in the MUCBL macro for #formats. The address that must be specified here is that from the 4th word in the address list (address of the terminal type) incremented by 7. This address list is provided by openUTM in the FORMAT user exit (see the openUTM manual “Programming Applications with KDCS for COBOL, C and C++”).

openUTM does not recognize when +formats are used in the FORMAT user exit. When data is transferred to the message area, only 8 bytes are cut off at the start of the conversation.

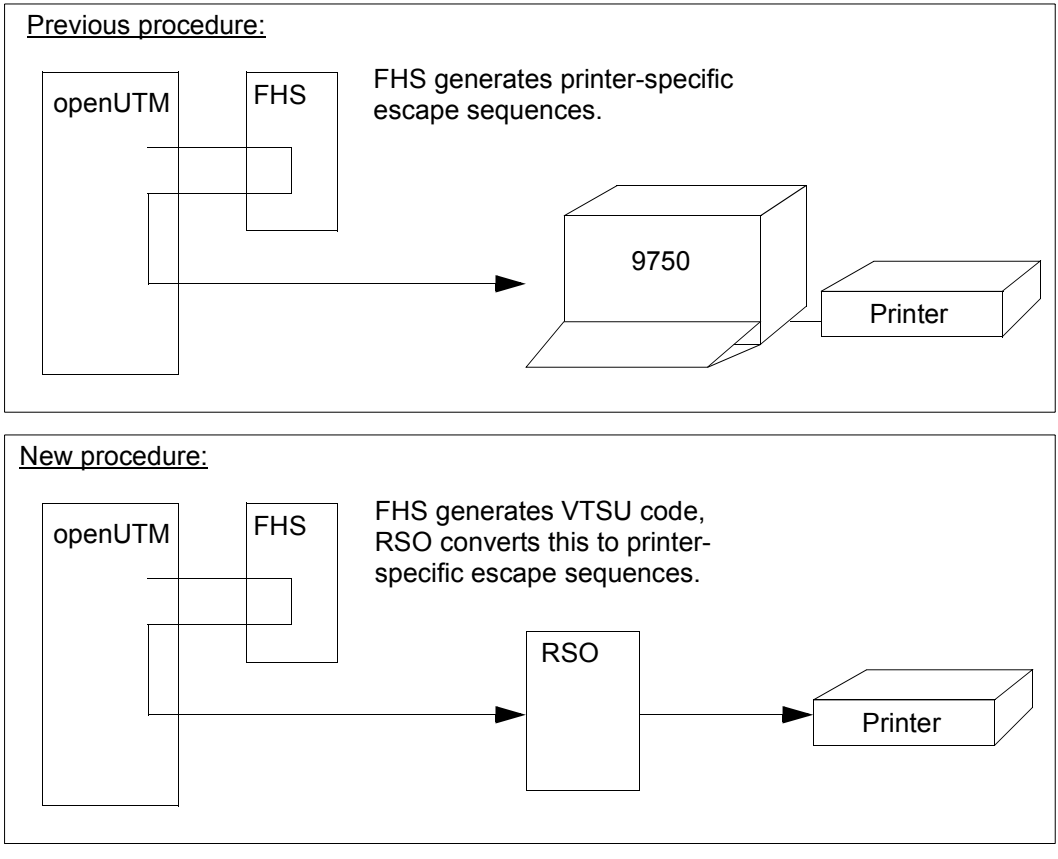
FHS provides the length fields ....MEAL and ....MUIL (see page 262ff) in the user’s own control block. One of the two lengths must be entered in the length field in the first word of the formatting user area (address in word 5 of the address list) in the form X’0000LLLL’. openUTM compares this length specification with the specification in the KCLA parameter. If it is greater than the value specified in the KCLA parameter, return code “01Z” is output.

The specification for the format library is independent of what is specified in the FHS start parameters.

The dialog extension (FHS-DE) can not be used in the FORMAT user exit.

5.1.9 External interfaces to RSO printers

Instead of generated printer-specific escape sequences, FHS produces logical VTSU code. This code is recognized by RSO and converted to printer-specific escape sequences that can be processed by the respective terminal.



The previous procedure continues to be supported. You define which procedure is to be used when you create a format in the IFG.

The application model

The printing of FHS formats from a UTM application was previously restricted to the printers connected to the terminal nodes of the TRANSDATA network, which could be accessed via the TRANSDATA-810 protocol.

Now, through using the OLTP interface between openUTM and RSO, UTM applications that use FHS V 8.3 can access all printers defined in RSO. This means that an UTM application can output an FHS format on any printer connected to a PC or the LAN, which has been configured in RSO.

Printing of Unicode formats is allowed only on printers defined in KDCFILE as RSO printers.

If you try to output a Unicode format on another printer, the return code FC37 is issued.

Advantages

- All printers are defined in RSO

  As soon as a printer is defined in RSO it is accessible for UTM applications for printing FHS formats. The prerequisite here is that you enter the device in the KDCFILE file used by the UTM application.

  You will find further information on declaring a printer in the openUTM manual “Generating Applications”.

- Central administration

  If the printers can be accessed from RSO and UTM applications, the type and the address of the respective printer must only be configured in RSO. The printer must simply be declared in openUTM (without specifying the type and address). If this printer is replaced by a printer of a different type, the KDCFILE used by openUTM must not be updated.

- Printers are recognized in RSO, but not in FHS

  Printer types that are recognized in RSO but not in FHS, are supported automatically.

  For example, FHS formats can be output on a printer that supports the IBM Proprinter-protocol. The printer must be configured in RSO for this purpose as a 9000-PRO-REMOTE-PRINTER and entered as an RSO printer in the openUTM KDCFILE.

- Printers will also be supported in the future in RSO

  Printers that will be supported in future versions of RSO are likewise supported automatically by FHS, assuming they are declared correctly in the openUTM KDCFILE.

Representation of field attributes

If a printer is recognized as an RSO printer, the following attributes will be used.

Printers recognized in IFG/FHS:

FHS attribute	Representation of screen attributes on the printers									
	9001	**9002**	**9003**	**9004**	**9011**	**9012**	**9013**	**9022**	**9001 -31**	**PCL**
NORM	normal	normal	normal	normal	normal	normal	normal	normal	normal	normal
BRT	-	-	-	bold	under-lined	bold	bold	bold	bold	bold
SIGN	-	-	-	shaded	italics	-	-	shaded	italics	italics
ITAL	under-lined	under-lined	red	under-lined	under-lined	under-lined	under-lined	under-lined	under-lined	under-lined

Note

The grayed out attributes behave differently if the printer is used with a BAM connection.
The WIDE and TALL attributes are not supported by RSO printers.

Printer types supported by RSO but not by FHS:

IFG/FHS Attributes	VTSU Code
NORM (normal)	NOR
BRT (bright)	EM3
SIGN (flashing)	EM1
ITAL (underlined/italics)	EM2

You will find a description of the physical representation of the VTSU code on the different printers in the manual “RSO V3.4A (BS2000/OSD)”.

The impact of preformatting on the behavior of FHS

Unicode formats may not be preformatted.

When you create a format in IFG, you are allowed prepare formats for output to specific printers. The result of this is that only parts of the message have to be generated dynamically in order to print this format with the path length to be followed in FHS being shortened.

The disadvantage of this method is that these formats can only be output uncorrupted on one device type and possibly on upward compatible devices.

The format was created in IFG	The printer is defined for openUTM	The printer type is known to FHS [1]	Behavior of FHS
with preformatting	as an RSO printer	Yes	The format is output on the printer[2]
		No	
	as a TRANSDATA printer	Yes	
		No	
without preformatting	as an RSO printer	Yes	FHS generates generic VTSU code and possibly printer-specific VTSU code
		No	FHS generates generic VTSU code
	as a TRANSDATA printer	Yes	FHS generates printer-specific Escape sequences[3]
		No	FHS generates error code [3]

[1] Printer types known to FHS:
9001, 9002, 9003, 9004, 9011, 9012, 9013, 9022, 9001-x31, PCL, 3287

[2] This behavior is the same as the behavior of previous versions of FHS.
A preformatted format is sent to the printer, regardless of whether the printer type is the same as the printer type for which the format was preformatted. The printout could be corrupted as a result.

[3] This behavior is the same as the behavior of previous versions of FHS.

5.1.10 Other notes

Loading P keys with FHS

You can also use FHS to load the programmable keys on your terminal. You have two options:

– output of the P key format as a “*” format with MPUT, in which case the follow-up program unit should not contain an MGET call.

– loading of the P keys with output of a “#” format, in which case the corresponding global attribute has to be set with the name of the P format. It is advisable to use this option only in the sign-on procedure because P keys are not reloaded in the event of a screen restart.

For further information on loading the P keys with FHS, see page 43.

UTM control field

UTM control fields may not be UNICODE fields.

Lowercase letters are only converted to uppercase in the UTM control fields if this is explicitly specified in IFG when a format is generated. openUTM always converts characters to uppercase in conventional TAC fields.

Using exit routines

Exit routines are used in the same way in UTM applications as in DCAM/TIAM applications (see page 286 for ASSEMBLER and page 391 for COBOL). The library that contains the exit routines must be specified in the FHS start parameter EXIT.

5.2 FHS dialog extension

Formats that conform to the Alpha Style Guide can be displayed on a terminal using the FHS dialog extension, or FHS-DE for short. You can thus program a user-friendly dialog for UTM applications and at the same time reduce the load on the UTM application. For further information on formats (= masks) that conform to the Alpha Style Guide, refer to the manual "Style Guide Guidelines on the Design of User Interfaces".

FHS-DE offers the following options:

– structuring formats in accordance with the Alpha Style Guide

– user-friendly dialog using selection fields, commands, and function keys

– multi-level intermediate dialog via dialog boxes

– application-specific help systems and messages

– extended input validation independently of the application

– simplified list output.

You generate the formats for FHS-DE using the format generator IFG V8.0 and later; see the "IFG for FHS" User Guide. These formats are hereafter referred to as FHS-DE formats, or DE formats if the context is unambiguous.

You can use existing IFG formats (up to IFG V7.1) for some FHS-DE functions, i.e. you can use these functions without modifying your application program. To do this, you must convert the formats in question before using them. For a description of how to convert these formats, refer to the "IFG for FHS" User Guide for IFG V8.0. For further information see page 168.

5.2.1 Structure of DE formats

You must create DE formats with IFG V8.0 or later. A DE format is a #format, i.e. a format with separate areas for global attributes, field attributes, and for field contents; see page 46ff.

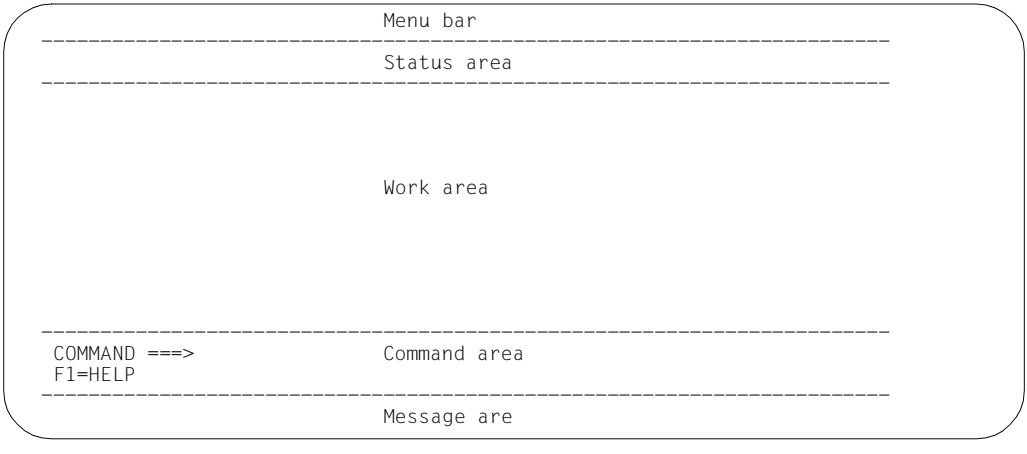
To receive a DE format, specify the switch “Dialog extensions required? : Yes” in IFG format 0706 (“User Profile/Format Display Attributes”).

There are two representations for a DE format:

– as a full format or partial format, i.e. the format occupies the entire width of the screen.

– in a box, i.e. the height and width of the format only occupies one part of the screen; see section “Dialog boxes” on page 109ff.

A DE format occupies additional areas that are used exclusively for user prompting. It is usually composed of four areas and has the following structure:



Menu bar (as of IFG V8.1)

The menu bar is a one-line area, delimited by a separator line, at the top margin of the screen. It contains menu titles. Each menu title represents a group of choices that are displayed in the form of a pull-down menu under that menu title. A menu bar is only possible in full-screen mode, not in a dialog box. No menu bar is permitted in partial formats.

Status area

The status area contains (centered) the title of the format (or panel). The format name (PANELID) is also output at the beginning of the line if requested with the FHS command “PANELID ON”; see the section on “FHS commands”. The title is optional and is defined using IFG format 0102. Formats without a title do not have a status area.

Work area

The work area is the actual action area for the terminal user and contains the text fields and variable fields (as for the existing FHS formats). FHS-DE may also have selection fields and list outputs.

Command area

The command area contains the command line, as well as the display (one or two lines long) of the function key assignment. The command line consists of a text field (in the example “COMMAND ===>”) and an input field for commands; this is known as the *command field*.

Message area

Messages are displayed in this area. The message area may only be defined for full or partial formats. If a partial format contains a message area, the partial format must be the format on the lowest part of the screen.
Boxes do not have a message area.

Long messages can also be displayed in special boxes, called message boxes; see page 153.

Of all the areas described, the work area is the only area which must be included for reasons of compatibility with earlier versions. It is strongly recommended, however, to define the DE format with all the permitted areas, so that the terminal user is presented with a uniform and user-friendly screen interface.

5.2.1.1 Global attributes of a DE format

The existing global attributes are retained for FHS-DE; see page 47ff. However, there are some new global attributes:

– The attribute group FORMATTING CONTROLS has been extended to include the attribute field for LANGUAGE EXTENSION.

– The attribute groups DIALOG PARAMETERS and DIALOG CURSOR POSITION are new.

Only the new global attributes are described below.

FORMATTING CONTROLS

These attribute groups have been extended.

LANGUAGE EXTENSION

Values:

A ... Z
A search is carried out in the format library for a member whose eighth character is the language extension specified here. The format name specified for MPUT should consist of a maximum of 7 characters. If it is less than 7 characters, it is filled with the character “#” by FHS.

Blank
FHS works without language extensions

X’00’
As for blanks

Further information on language extensions is contained in the section “Language extensions” on page 162.

DIALOG PARAMETERS

MESSAGE IDENTIFICATION

Code for the message which is to be output in the specified format. When defining the message, it is specified whether this message is to be output in the message area of the format or in a message box.

MESSAGE LOCALIZATION

This attribute applies for message output to a message box, and it determines the reference point (upper left corner) of the message box.

Values:

Field name
Reference point in the specified field.

$lll#ccc
Absolute specification of the reference point: line number lll, and column number ccc; ccc and lll are three-digit values.

Blank
Reference point is the cursor position defined with DIALOG CURSOR POSITION; see below.

FHS positions the message box '+2/+2', i.e. 2 lines below and 2 columns to the right of the reference point. For further information, refer to the section "Implicit boxes" on page 118.

The addressing aid contains another field for this attribute group. This field (MESSAGE INDEX) is reserved for later versions.

For further information on message output and message boxes, see the section "Outputting messages" on page 153.

DIALOG CURSOR POSITION

Z-CURSOR FIELD

This attribute field can be used for input and output:

Input:
For input, the name of the field which contains the cursor is supplied; the name is the field name defined when the format was created with the IFG. If the cursor is in an unnamed field, e.g. a text field, FHS supplies the absolute screen position of the cursor in the form “$lll#ccc”. Here, lll is the number of the line, ccc is the number of the column, e.g. $010#032 for line 10, column 32. lll and ccc always have three characters.

Output:
Specifies the position of the cursor for format output. In this case, either the name of the field in which the cursor is positioned or the absolute position of the cursor in the form or the absolute position of the cursor in the form If a field name is entered, then the cursor can be positioned anywhere within that field by means of the global attribute Z-CURSOR POSITION; the global attribute Z-CURSOR INDEX is also evaluated.
If nothing is entered in the global attribute Z-CURSOR FIELD, then the entries of the global attributes CURSOR CONTROL and CURSOR POSITION apply (as before, see page 47ff).
For entries that have errors, e.g. incorrect field names, the UTM application aborts the process and reports “formatting error”.

Z-CURSOR POSITION

Z-CURSOR POSITION specifies the character – Unicode character or 7-/8-bit character depending on the field type – in the field where the cursor must be set if the cursor position is within a named field, i.e. if Z-CURSOR FIELD contains a field name. This attribute is evaluated at input and output.

Permissible values: 0 or 1 through field length (0 is treated as 1 and corresponds to the first position).

If Z-CURSOR FIELD does not contain a field name, 0 is always entered at input.

Cursor index (Z-CURSOR-INDEX)

The entry in Z-CURSOR-INDEX depends on the contents of the Z-CURSOR-FIELD and has the same meaning at input and output:

– If the name of a field that belongs to a list line is in the Z-CURSOR-FIELD, then Z-CURSOR-INDEX contains the line number of the list. Possible values: 0 or 1 to the number of existing list lines. (A list line may consist of multiple screen lines.)

– If the name of a field that appears in several partial formats under this name is specified in Z-CURSOR -FIELD, then Z-CURSOR-INDEX describes the number of the field in question.

– In all other cases Z-CURSOR-INDEX must contain the value 0.

5.2.1.2 Field contents of a DE format

The following extensions exist for DE formats compared with previous #formats:

– The addressing aid of a DE format can also contain a command field; see the section “Commands” on page 131. The command field is always addressed with the name CMDAREA when positioning the cursor, even if IFG defines a different name.

– For DE formats with single-choice selection fields, the addressing aid has an area for exclusion tags; see the section “Selection fields” on page 124. There is no field which corresponds to this area on the screen.

– The addressing aids for list areas also contain an area for control information. Included in this area are, e.g. scroll information or information on modified lines; see the section “Outputting lists” on page 127ff. This control information does not have a corresponding input field on the screen.

5.2.2 Menu bar and pull-down menus

Formats may include a one-line menu bar in which menu titles are shown. A menu title is selected by moving the cursor to that menu title and pressing the Enter key.
No menu bar can be defined in partial formats.

```
 File      Project     Help
--------------------------------------------------------------------------------
         : _ 1. Add         : inistration - Logon
         :   2. Delete      :
         :   3. Print...    :
 Please e:   *. View        :  in the appropriate fields.
         :...................:

 Date               : 02.12.1994

 Project name       : .....................................

 Author             : .....................................

 Version            : ................

 Programming system : ...........




 COMMAND ===> ................................................
 F1=Help  F3=End  F12=Quit
```

Working with pull-down menus

Pull-down menus are only possible in full-screen mode. The menu bar and the pull-down menus are generated using the Interactive Format Generator IFG. The menu titles in the menu bar are markable input fields, but the inputs and markers of menu titles are ignored. Selection occurs solely on the basis of the cursor position. In the addressing aid, menu titles are the first fields of the mask. A menu title is selected if the value DETECTED is entered in the associated INPUT_STATE_ACT field attribute. This value is set only if control is returned to the application directly after a pull-down menu is displayed. FHS-DE ensures that this value is only shown for one menu title.

The entries made in the pull-down menu are transferred to the input fields. These entries do not occur if the ACTIONS command was given following input in a pull-down menu or if some other pull-down menu was selected.

A selected menu is recognized only by the DETECTED field attribute of the associated menu title in the data transfer area.

Positioning the cursor in the menu bar

One method of placing the cursor in the menu bar to press the F key that was assigned to the system command ACTIONS (F10 by default).
The ACTIONS key initiates cursor positioning by FHS-DE. The current position of the cursor is saved, and the cursor is moved to the first character of the first menu title of the menu bar. When the ACTIONS key is pressed again, the cursor is reset to the saved initial position. This method of positioning the cursor requires a data transfer to the processor.

Another method is to use the cursor keys. Since this function is implemented by hardware, the cursor position cannot be saved by FHS-DE in this case.

Selecting a menu title

Within the menu bar, you select a menu title by moving the cursor with the Tab key or the cursor keys to a character of the desired menu title and by pressing the Enter key. This causes a pull-down menu with the available choices for that menu title to be displayed. During the period that the pull-down menu is shown, the cursor is positioned within the choice fields of that menu, and all fields of the underlying mask are locked (i.e. cannot accept inputs).
Markers or inputs in the menu title are ignored, i.e. cannot be used to select another pull-down menu.

Activating a selection in a pull-down menu.

The pull-down menu contains a single-choice selection field. The selection is activated by entering a choice in the selection input field or by using the default value and pressing the Enter key when the cursor is in the pull-down menu. The value is then entered into the data transfer area, and control is returned to the application program.

Displaying another pull-down menu

You can switch to another pull-down menu by selecting some other menu title with the cursor keys.

The CANCEL key can also be used to return to the menu bar when a pull-down menu is displayed, but the pull-down menu is deleted in this case, and any selection entered in it is ignored. The cursor is positioned at the first menu title in the menu bar, and any of the menu titles may be then selected.

Canceling a pull-down menu and exiting the menu bar

If the ACTIONS key was used to enter the menu bar, you can cancel the displayed pull-down menu by pressing the same key again. The pull-down menu is then deleted, and the cursor is reset to the saved position in the work area. Any selection made in the pull-down menu will be ignored. The CANCEL key may also be used to cancel the display of a pull-down menu. In this case, however, the cursor is moved to the first menu title of the menu bar, and you can then exit the menu bar by using the cursor keys.

CAUTION!

Make sure you define an ACTIONS or CANCEL key for a format with a menu bar! A pull-down menu is not assigned a separate key, and no command can be entered in the command input field when it is displayed. In other words, it is not possible to exit the pull-down menu without an ACTIONS or CANCEL key.

5.2.3 Dialog boxes

You can implement intermediate dialogs with FHS-DE by overlaying the underlying format with dialog boxes; see the manual “Style Guide”. The intermediate dialog can have multiple layers, i.e. several boxes can overlay each other on the screen.

A dialog box is a frame in the form of a ’screen within a screen’. This frame contains a format that no longer covers the entire area of the screen. You can generate the formats displayed in boxes by means of the IFG format generator.

A box contains the following:

– a format with input, output and selection fields
– messages, see the section on “Message boxes”
– help texts, see the section on “Help boxes”.

Some boxes are used solely for information output and require no input, e.g. certain help or message boxes. These boxes are referred to as *modeless* boxes. Further entries may be made in the underlying areas of the screen.

Modal boxes expect input. Once a modal box is output, all the other sections of the screen are protected against input.

With FHS-DE, boxes can be output by means of the program unit or by FHS with FHS-DE. In the case of output with the program unit, the boxes are *explicit* boxes. Boxes output by FHS (independently of the partial format) are called *implicit* boxes. Help texts are typically output as implicit boxes, i.e. you can create a complete help system without increasing the application load.

Explicit boxes are output using MPUT calls in the program unit. For information on programming these MPUT calls see the section “Explicit boxes” on page 112ff.

The following is a simple example of full format with a dialog box.

```
                    Address management - new entry
------------------------------------------------------------------------------

 Please enter the address data in the corresponding fields..


 Last name: Smith .--------------------------- First name: Roger.............
                  | Professional status      |
 Street:  Northsi |                          |
                  | Please select:           |
 Zip code: 47401  | _  1. Salaried employee  | Bloomington.....................
                  |    2. Wage earner        |
 Tel: 0711-9905   |    3. Self-employed      |
                  |    4. Training           |
 Professional: *  |    5. Other              | Marital status: married
 status           |                          |
 Children: 01     | F1=HELP  F12=EXIT        | Denomination: protestant
                  ---------------------------


------------------------------------------------------------------------------
 Command:
 F1=HELP  F3=EXIT  F12=CANCEL
------------------------------------------------------------------------------
```

During the first dialog step, the terminal user entered nothing in the field ’Professional status’ output on the screen with the default ’*’. The program unit of the second dialog step interprets the ’*’ character appropriately and displays the box for the selection field ’Professional status’.

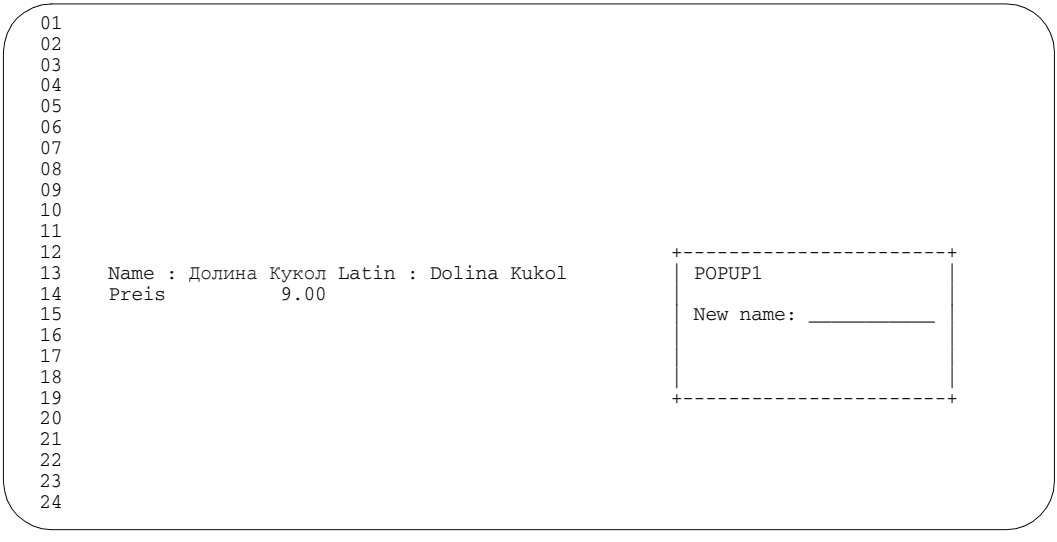
Entries in dialog boxes

If, (as in the example) one or more boxes appear on the screen, in addition to the basic format, then it is only possible to make entries in the uppermost modal box. All modal boxes/formats below this are inactive, i.e. the input fields become protected fields and are displayed accordingly on the screen. An underlying box will only be activated after all the overlying boxes have been removed.
If the uppermost box is modeless, e.g. a help box for a specific field, then it is possible to make entries in the underlying box/format, as long as the required fields are not completely or partly concealed. For input in concealed fields, the box must be first removed.

As soon as one format or one popup containing at least one UNICODE field has been output on the data display terminal, this format and all formats resulting of this one and overlaying popups or messages is output in Unicode mode. You are able to type Unicode characters in all input fields of the overlaying popup. This results from the fact that the

Unicode mode of the terminal emulation is valid for the whole format image on the screen. At input time, FHS will check that the characters that you enter in the fields are compatible with the character set of the format.

Example



In this example, a background format is output on the data display terminal. This format contains at least one UNICODE field (the “Долина Кукол” string). At a further dialog step, a dialog box (POPUP1) is output over this background format. The format resulting of the merge of the background format and the popup must be output in Unicode mode; else the emulation is not able to display the “Долина Кукол” text correctly. That means that you are able to enter any Unicode character in the input field of the popup, even if this popup was not primarily designed to receive Unicode characters. If you enter characters that are not compatible with the base character set of the format, FHS will issue an error message. See also chapter “Structure of the data transfer area” on page 45.

Refer to the “Unicode in BS2000/OSD” introduction for a survey of the Unicode support in the BS2000/OSD as well as basic information on Unicode.

Removing dialog boxes

Implicit boxes are removed using the FHS commands CANCEL and EXIT. CANCEL removes the uppermost implicit box and EXIT removes all other implicit boxes; see the section “Commands” on page 131.

Explicit boxes can only be removed by the program units of the application.

For further information on working with boxes, see the section “Information for the terminal user” on page 166.

5.2.3.1 Explicit boxes

Explicit boxes consist of a frame that contains a DE format. They are output with MPUT calls. The following options are available to the programmer:

– define the position of a box (ADDPOP function)
– remove boxes (REMPOP function)
– replace boxes (remove and output again)

Explicit boxes must always be removed by the program unit; it is not possible to remove explicit boxes using FHS commands (unlike implicit boxes).

Two MPUT calls are usually required for handling boxes. The first MPUT supplies the parameter for the box, e.g. the initial position of the box. The second MPUT displays the actual format.

The parameters for the box are supplied in a separate area, the POPUP control block, or POPUP-CB for short. A data structure of the POPUP-CB is available in the SYSLIB.FHS library for the relevant programming language. These data structures can be copied into the program and are listed on page 173.

A box is output in three steps:

Step 1: define parameters in POPUP-CB

Step 2: MPUT with KCMF = #!POPUP and NB = POPUP-CB

Step 3: MPUT with KCMF = #format name and NB = transfer area

You must specify the POPUP-CB as a message area for the first MPUT. Specify the pseudo format #!POPUP as a format name. This format name is reserved and informs FHS that the MPUT call refers to a box. The second MPUT is a normal MPUT call for a DE format; see the openUTM manual “Programming Applications with KDCS for COBOL, C and C++”.

If one or more boxes are removed and if the underlying format/box is displayed without modifications, then there is no second MPUT.

You can combine both options with corresponding entries in the POPUP-CB, i.e. you can replace boxes by removing one or more boxes and simultaneously outputting another box.

The following diagram shows how to supply the POPUP-CB and how to program the first MPUT call.

Supply the parameters

Contents	Field name in POPUP-CB	
“R” / 0 /blank	POPUP-REMPOP	1.
“A” / 0 / blank	POPUP-ADDPOP	2.
Number of boxes / 0	POPUP-RM-LEV	3.
Line spacing	POPUP-AP-LINE	4.
Column spacing	POPUP-AP-COL	5.
Field name / 0 / blank	POPUP-AP-NAME	6.

Contents	Field name in KDCS parameter area	
“MPUT”	KCOP	7.
“NT”	KCOM	7.
“24”	KCLM	8.
blank	KCRN	9.
#!POPUP	KCMF	10.
0	KCDF	11.

KDCS call

1st parameter	2nd parameter	
KDCS parameter area	POPUP-CB	12.

Return information from openUTM (as for other MPUT calls)

Enter the following parameters for the POPUP-CB:

1. in the **POPUP-REMPOP** field, the code for removing the boxes:

R	Remove box(es). In the POPUP-RM-LEV field, define how many boxes are to be removed.
binary 0	No boxes to be removed
Blank	As for binary 0

2. in the **POPUP-ADDPOP** field, the code for creating boxes:

A	Create box
binary 0	No boxes to be created
Blank	As for binary 0

   If you supply both POPUP-REMPOP and POPUP-ADDPOP with binary 0 or a blank, the position of an existing box is reused; see “Features of explicit boxes” on page 116.

   If you supply POPUP-REMPOP with R and POPUP-ADDPOP with A, then one box (or more) is replaced by another.

3. in the **POPUP-RM-LEV** field, the **number** of explicit boxes that are to be removed. If you enter 0, all existing explicit boxes are removed. The **number** should not exceed the number of boxes on the screen, otherwise there are formatting errors.

   POPUP-RM-LEV is only evaluated if POPUP-REMPOP = R.

4. in the **POPUP-AP-LINE** field, the **line spacing** between the starting point and the point of reference. The starting point is the first character (top left) of the format in the box. POPUP-LINE is only evaluated if POPUP = A:

+n	the starting point is n lines below the reference point
-n	the starting point is n lines above the reference point

   The reference point is defined in POPUP-AP-NAME: if POPUP-AP-NAME contains binary 0 or blanks, the top left-hand corner of the underlying box/format is taken. If POPUP-AP-NAME contains a field name, this field is the reference point.

   For further information on positioning a box, see “Features of explicit boxes” on page 116.

5. in the **POPUP-AP-COL** field, the **column spacing** between the starting point of the format in the box and the reference point. POPUP-AP-COL is only evaluated if POPUP-ADDPOP = A:

   +n the starting point is n columns to the right of the reference point

   -n the starting point is n columns to the left of the reference point

   If both the line spacing and column spacing are equal to 0, the box is positioned by +2/+2 on the reference point.

6. in the **POPUP-AP-NAME** field, the **reference point** for the box:

   Field name
   Input field in the underlying format

   binary 0
   Top left-hand corner of the underlying format

   Blank As for binary 0

   POPUP-AP-NAME is only evaluated for POPUP-ADDPOP = A.

For the MPUT call, enter the following in the KDCS parameter area:

7. in the fields **KCOP** and **KCOM**, the values **MPUT** and **NT** (for message segment)
8. in the **KCLM** field, the length of the POPUP-CB (24 bytes).
9. in the **KCRN** field, **blanks** because the box is output on a terminal.
10. in the **KCMF** field, the format name **#!POPUP**, because a box is to be output.
11. in the **KCDF** field, the value **0**, because the box is a #format.

For a KDCS call please enter:

12. as 1st parameter, the address of the KDCS parameter area.
    as 2nd parameter, the address of the POPUP-CB.

Features of explicit boxes

- Position and size of a box

  The *position* of a box is defined by the entries in POPUP-AP-NAME, POPUP-AP-LINE, and POPUP-AP-COL.

  Example

  If the format (in the box) is to begin exactly three lines below the ‘PROFESSION‘ field, then you must enter the following: POPUP-AP-NAME=PROFESSION, POPUP-AP-LINE=+3 and POPUP-AP-COL=0.

  The *size* of a box is defined by the format specified in the second MPUT call. However, it can happen that the space at the specified position is insufficient. In such a case, FHS reacts as follows:

  – If you have explicitly specified the position of the box, i.e. if you have supplied at least one of the fields POPUP-AP-LINE or POPUP-AP-COL, then FHS reacts with a formatting error. openUTM thus aborts the conversation with PEND ER.

  – If you have not explicitly specified the position of the box, then FHS first tries to position the box with the default movement (+2/+2). If this is not possible, FHS moves the box so that it fits on the screen. If the reference point is a field, the field remains visible if possible, i.e. FHS tries to output the box below, above, or to the right of the field.

- Remove boxes

  – If you want to remove boxes without outputting another box at the same time, then *one* MPUT is sufficient with the POPUP-CB. There is no need for the second MPUT with the DE format.

    After the PEND of the program unit, the required number of boxes is removed. For the subsequent INIT call, openUTM supplies the format ID of the uppermost remaining box (i.e. not #!POPUP). If all the boxes have been removed, the INIT call supplies the ID of the basic format.

  – For reasons of performance, you should design the application in such a way that boxes lying on top of each other are not removed individually.

  – *All* boxes are removed if only the (second) MPUT call is specified with the format but no MPUT is specified with the POPUP-CB is not specified. FHS then executes an internal REMPOP and outputs the required format.

Reuse the same position for a box

A box position specified with ADDPOP can be used as follows for outputting further boxes:

First output:
1st MPUT: Supply POPUP-CB completely (as described above).
2nd MPUT: KCMF=#format1

Second output:
1st MPUT: POPUP-ADDPOP = 0 and POPUP-REMPOP = 0
2nd MPUT: KCMF=#format2.

“format2” need not be equal to “format1”, but “format2” must fit this position; otherwise a formatting error occurs.

Further outputs similar to the second output are possible.

- Replace boxes

  You can replace boxes by entering POPUP-REMPOP = R and POPUP-ADDPOP = A in the POPUP-CB for the first MPUT call, and output a format with the second MPUT. FHS removes the required number of boxes (entry in POPUP-RM-LEV) and generates the required new box. Define the position in POPUP-AP-NAME/POPUP-AP-LINE/ POPUP-AP-COL. The reference point for the new box is the box (or the format) that remains after removal of the uppermost box.

- Attributes of explicit boxes
  - If formats for 8-bit terminals have been defined, then all formats which are to be overlaid, should have the same CCS name; see “Code name” on page 501.
  - Overlaid formats always have the same attributes as the current box.

5.2.3.2 Implicit boxes

Implicit boxes are controlled by FHS and are used to output messages and help panels. In the case of implicit boxes, the program unit knows neither the name nor the fields of the format.

Implicit boxes are output by FHS as modal or modeless boxes. For modal boxes, the underlying format is inactive, whereas for modeless boxes, it is possible to make entries if the relevant field is not concealed.

Position of implicit boxes

The position of implicit boxes is defined by FHS. FHS displays boxes that do not have a reference point in the middle of the screen. FHS first tries to display boxes that have reference points (e.g. field-related help or messages) by means of the default movement, i.e. two lines below and two columns to the right of the reference point. If there is not enough space, FHS tries to output the box in such a way that the reference field remains completely visible. If, however, the field is partly or totally concealed, then the reference field also becomes a protected field for modeless boxes.

Attributes of implicit boxes

If an implicit box overlays the entire screen, the attributes of the implicit box apply; the default values apply for global attributes.

If the underlying format remains partly visible, then the implicit box receives the attributes of the underlying format. FHS does modify some attributes and sets them as follows:

```
INIT CONTROL          →   'FIRST INIT'
TABULATOR CONTROL     →   'NO AUTOTAB'
CURSOR CONTROL        →   'DEFAULT'
P-KEY-SET             →   ' ' (=default character)
ALARM CONTROL         →   'DEFAULT' or
                          'A' (only for message boxes for serious errors)
```

If a CCS name is specified in a help panel, this must match the CCS name of the underlying format.

5.2.3.3 Message boxes

A message box is an implicit box and is generated by FHS. The height and width of the message box depend on the length of the message text. If the specified maximum size of the box (six lines of 56 characters each) would be exceeded as a result of editing, the message text is transferred to the message box in unedited form.

A message box is structured as follows:

```
.....................................................
: Message code                    : Message identification
: Message text                    :
:          .                      :
:          .                      : Message text
:          .                      :
: Message text                    :
: ==>                             :  Command line
: F1=Help   F3=...                : Key assignment
.....................................................
```

The text of the message is highlighted, e.g. bright, reverse video, or in color, depending on the terminal and the type of message. The other areas of the box are displayed in grayed-out mode. The message code only appears when the PANELID ON command has been specified beforehand.

The command line is only generated if one of the FHS commands CANCEL or EXIT (or HELP, if help on the message was agreed) is not available, i.e. if the associated F key is assigned by the UTM application.

As of IFG 8.1 the prompt text is entered in the help format.The setting for the default format is used only if the specification is not present in the format.

A message is output either by FHS (implicitly) or by the program unit (explicitly). The program unit can suggest a position in the global attributes for implicit messages; see the section section “Global attributes of a DE format” on page 102.

For implicit messages, the message code always takes the form IDHSnnn; for explicit messages, the message code is entered in the global attribute MESSAGE IDENTIFICATION of the underlying format; see the section “Outputting messages” on page 153.

The size of the message box is determined by FHS from the scope of the message to be displayed. Codes to edit the message can be specified when creating the message text.

5.2.3.4 Help boxes

A help box is a modeless implicit box. The size of application-specific help boxes are defined when generating them with IFG; they can cover the entire screen.

Field-related help boxes should be smaller. In this case, FHS outputs them underneath the field if possible; see the section on “Position of implicit boxes” above. If, as a result, the field remains visible, you can read the help and fill the field at the same time.

The size of a help box is defined by the help panel. In the definition of the help panel you can specify whether a help box is to have a specific height or whether FHS-DE may increase the height of the box depending on its position. In the latter case, the defined height is then the minimum height of the box. The message area of the screen is not overwritten when extending the box, and the width of the box is not changed.

If a help text does not fit in a box, the box contains a page-turning information, e.g. “More: +”. You can request other texts using the command “+”. You can change the prompt “More:” by modifying the default format “IDHSCRLx” with IFG. In this case, x is the language extension; see page 162.

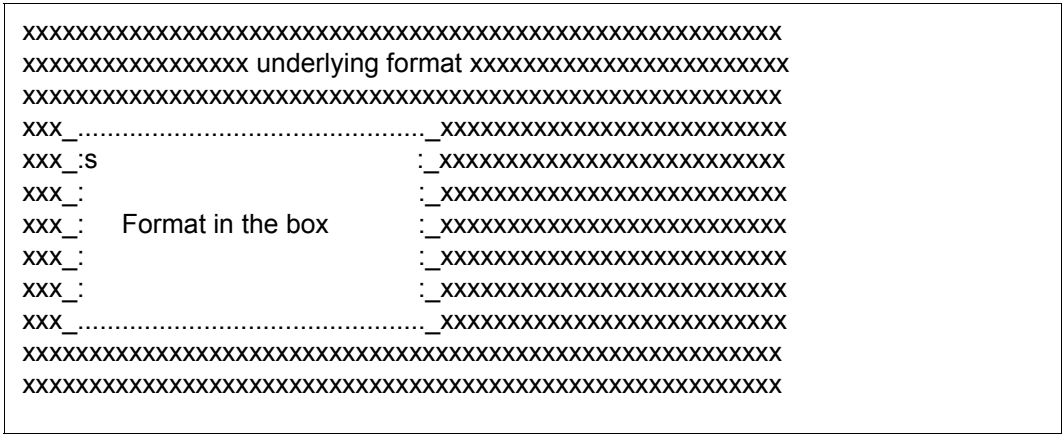
You generate a help box as a DE format with IFG. You must explicitly specify that the format is a help panel.

Further information on help can be found starting on page 156.

5.2.3.5 Frame of a box

The frame of a box is created with the normal character set. It is composed of periods (horizontal lines) and colons (vertical lines). To change these characters, see “Modifying frame characters” on page 121.

FHS adds blanks in certain places between frames and the underlying format. The following diagram shows how a format fits in a box and how the box is positioned on the underlying format.



'_' stands for a blank or an attribute character in the case of a 3270 terminal, and 's' indicates the starting point of the format in the box.

The first and last column of the format in the box should contain blanks.

Modifying frame characters

An application can use other characters for the frame. In order to do this, you must change the macro operands of the CSECT IDHBORD in your source program and then generate a new IDHSDEV object module. IDHSDEV contains the macro IDHBDR, which is used to define the frame characters. The macro may be coded more than once to define different frames for different terminals. The IDHSDEV entry is IDHBORD.

IDHSDEV must be contained in the library that contains the modules of the formatting system. IDHSDEV can be statically or dynamically linked to the application program; however, the entry must remain external so that FHS can access this entry if FHS is dynamically loaded. If the module IDHSDEV is not found, the frame is composed of the default characters (= periods and colons; see above).

```
IDHSDEV      START

IDHBORD      CSECT
             IDHBDR parameter
             END
```

IDHBDR has the following parameters:

Operation	Operands
IDHBDR	[NEXT=name] [,DEV=device] [,DIM=dimension] [,CCSNAME=ccsname] [,COLORED=color-screen] [,COLOR=color] [,BORDER=frame-characters [,ICENAME=icename]

Meanings of operands:

DEV, DIM, CCSNAME, and COLORED are used to select the frame definition.

BORDER=frame-characters

sequence of 8 characters that define frame characters:

1st character = top left-hand corner
2nd character = top right-hand corner
3rd character = bottom left-hand corner
4th character = bottom right-hand corner
5th character = upper horizontal line
6th character = lower horizontal line
7th character = left vertical line
8th character = right vertical line

ICENAME=icename

Name of an ICE character set (only for DSS9763).

The following ICE character sets are included in Version V08.1A:

IDHCSD1B (corresponds to IDHCHSET in FHS-DE V08.0A)

IDHCSD1C (good for 9763 color screen with black background)

IDHCSD1D (good for 9763 color screen with color background)

IDHCSD4B (9763, dimensions 27*132, monochrome)

Example

IDHDBDR BORDER=><><-->< defines dashes for the horizontal lines and angle brackets for the vertical lines.

If the IDHBDR macro was specified without parameters, the DSECT BORDER is generated.

The default frame characters used are '..::..::' * (period/colon).

If FHS-DE is to operate a 9763 data display terminal with a color screen, the following must be observed when preparing the frame definition:

1. When specifying the ICE color character set, white must be given as the COLOR operand.
2. The representation of color is determined by the device setting. SIDATA can be used to set whether characters in the specified color are shown on a black background or whether black characters are shown on a color background. In the latter case, inverse colors are used at the 9763 terminal, e.g. yellow frame characters become blue and cyan becomes red. For this reason, two color character sets for the frame definition are included in the delivery package: IDHCSD1C should be used for a black background, and IDHCSD1D for a color background. The best layout is obtained when cyan characters are displayed in low intensity (SIDATA setting). Please do not use the specification "HOLE COLOR: WHITE". with the format definition.
   Since the desired screen representation cannot be known in advance, no color character set is contained in the standard frame definition. An appropriate macro call has therefore been included as a comment in the following listing of the standard frame definition. If required, the Assembler source code IDHSDEV can be modified as needed and reassembled.
3. If you define a monochrome character set, it will be displayed in white on a background of the specified color or - if SIDATA was used to set the screen to a color background - in black on the complementary color.

Character sets can be defined by using the SNI product 'Interactive Character Set Editor (ICE)'.

Note

Some PC emulations for the 9763 terminal may return incorrect values for loadable character sets and may hence not produce the desired frame when frame characters are used with ICE character sets.
In such cases, you will need to modify the standard definition of the frame characters by not using ICE names for the 9763 terminal or by removing the macro call for 9763.

5.2.4 Selection fields

A selection field provides the terminal user with a simple way of choosing from a number of options. There are two types of selection fields:

– single-choice field: the terminal user selects *one* of several options.

– multiple-choice field: the terminal user can select *several* entries.

A selection field is part of the work area of a format and usually has a number of lines. It consists of a header, a series of entries, and one or more selection input fields.

You can assign a help panel (defined when the format is created using IFG) to both the selection field and each entry.

5.2.4.1 Single-choice field

A single-choice field always has *one* input field where the terminal user enters the character for selection. The input field can be one or two bytes in length. In accordance with the Alpha Style Guide, digits should be used for selection: one digit if there are up to 9 options, and two if there are 10 or more options.

You create a single-choice field with IFG by entering the corresponding special character at the beginning of the desired line. The following is an example of a single-choice field:

```
            Professional status

         Please select:
         _  1. Salaried employee
            2. Wage earner
            3. Self-employed
            4. Training
            5. Other

        Command:
        F1=HELP   F12=EXIT
```

The options are numbered 1 through 5 in this example. The user enters the appropriate number in the input field. Here the input field is marked with an '_' (underscore). The length of the input field and the character for selection are defined when the format is generated using IFG.

FHS checks the input. If it is correct, FHS transfers the character(s) possibly together with further input data - to the program unit in the MGET call. From the input the program unit must perform the corresponding processing steps.

If the input field does not contain an entry, no selection information is transferred to the program unit.

The program unit can prefill the input field with a default value or can lock certain options. The default value may also already be defined in IFG.

To *lock* the options, the data transfer area contains a special field. This field has one byte per option. The bytes appear in the order in which the options were defined using IFG.

An option is locked if the associated byte contains the digit ’0’ (i.e. X’F0’). If, for example, the third option is to be locked, the program unit must enter ’0’ in the third byte. If an option is locked, an asterisk (’*’) appears by default on the screen instead of the digit.

Besides the externally displayed value for each choice, a so-called “internal choice number” can be defined in IFG. This internal choice number is then returned to the application instead of the external one. If internal choice numbers have been defined, two characters are always reserved in the data transfer area, regardless of whether or not the length of the input field in the mask is one or two characters.
Advantage: if the external choice number needs to be changed, there is no need to modify the application code if the internal choice number is retained.

Selection fields in pull-down menus are defined as single-choice fields. These fields are placed at the end of the data transfer area (see “Addressing aid”).

5.2.4.2 Multiple-choice field

A multiple-choice field contains an input field for each option. The user selects an option by marking the relevant input field.

You create a multiple-choice field with IFG by entering the corresponding special character at the beginning of the desired line. The following is an example of a multiple-choice field:

```
     Childhood illnesses


        Please enter:
          _ Mumps
          _ Whooping cough
          / German measles
          _ Scarlet fever
          / Measles
          _ TB
          * Other illnesses

         Command:
         F1=HELP   F12=EXIT
```

The user selects the option by marking it with ’/’; in the example “German measles” and “Measles” are selected. The option “Other illnesses” is indicated as locked.

FHS checks the entries. If they are correct, FHS transfers the information to the program unit (MGET). The value ’1’ is supplied in the relevant data fields for each marked field. From this entry the program unit must perform the corresponding processing steps. If no option is selected, all the field attributes for the input status are set to NOT TOUCHED; the field contents are the same as for the previous MPUT.

The following entries are permitted in the program unit for the individual fields of the selection field:

binary zero or blank	selectable, not prefilled
’1’	preselection
’0’	selection blocked

The defined marker character (/ or x is recommended in the SNI Alpha Style Guide) appears on the screen if an option is preselected. The terminal user can overwrite this character with NULL or a blank. A locked option is marked on the screen with the exclusion character (default: ’*’).

5.2.4.3 Changing marker and exclusion characters

As the default option, FHS uses the character ’*’ for locking (exclusion character) and the character ’/’ for marking selection fields. These characters are defined in the formats IDHSCHCx (marking) and IDHSCHDx (excluding), where x is the national language identifier; see the section “Language extensions” on page 162.

You can change both of these formats with IFG and thus define other marker or exclusion characters; e.g. ’+’ for marking and ’-’ for excluding. Unicode characters are not allowed for marking and for excluding. In addition to the predefined characters FHS also always accepts the characters ’/’, ’X’ und ’x’ for selecting an entry.

5.2.5 Outputting lists

FHS-DE offers a simple option for outputting a large number of data records in the form of tabular lists and for modifying these records. To do this, you can generate “list areas” within a format by means of IFG; see the “IFG for FHS” User Guide.

The following options are provided for the list areas:

– Several data records of the same type are displayed in table form with a column title. The number of data records displayed is variable.

– The program unit is informed about all modified data records.

– Increased convenience for terminal users: you can scroll forward and backward.

You can create a list area using IFG and specify the characteristics of your list, e.g. column titles. The recommended list header in the SNI Alpha Style Guide is as follows:

from line xxx to line yyy out of zzz lines.

The following example shows how a list can be structured.

```
        List of items             Line 430 to 432 of 432   More: -

Number             Item
                                          Quantity              Price
101999388          M6 screw.....................................
                                             100              24,50
101227288          M6 nut.......................................
                                             100              18,70
101227335          40x40 square.................................
                                              10              16,30
****************************** END OF DATA ******************************

--------------------------------------------------------------------------------
 Command:
 F1=HELP  F3=EXIT  F12=CANCEL
--------------------------------------------------------------------------------
```

“List of items” is the title of the list; “Number”, “Item”, “Quantity” and “Price” are the column titles. Column titles may consist of up to 3 lines.

The scroll information “More: -” tells users that in this case they can only scroll back because they have reached the end of the data. The last line indicates this explicitly.
As of IFG Version 8.1, three variable output fields (430, 432, and 432 in the example above) can be defined in the list title. These variables show the line numbers of the first and last

line displayed on the screen and the overall dimensions of the list. FHS-DE cannot determine these values independently; they must be supplied in the data transfer area by the application program.

A list line consists of two lines in this example. Input fields here are the fields in the columns “Quantity” and “Price”. The user can modify these fields; all other fields are preset by the program unit.

The maximum number of list lines is defined when the format is created using IFG. The program unit can, however, output fewer list lines and thus fewer data records in a format.

List handling in the program unit

Similar to other DE formats, formats with a list area are read with MGET and output with MPUT. If you wish to scroll through the lists, you must run the program unit several times.

The addressing aid of a list area contains additional fields that control the output and input of the lists.

Addressing aid of a list area

The addressing aid of a list area contains some additional fields and has the following structure:

<table>
<tr><td colspan="3">Global attributes</td></tr>
<tr><td colspan="3">Field attributes for output fields in list title (line numbers, list size)</td></tr>
<tr><td colspan="3">Field attributes for all fields of a list line x max. number of list lines</td></tr>
<tr><td colspan="3">Field contents for output fields in list title (line numbers, list size)</td></tr>
<tr><td>Line field</td><td>Scroll info</td><td>MODINDEX area</td></tr>
<tr><td colspan="3">Field contents</td></tr>
</table>

The global attribute DIALOG CURSOR POSITION contains the current position of the cursor in the lists; the global attribute Z-CURSOR-INDEX contains the number of the list line in which the cursor is positioned. For further information, see the section “Global attributes of a DE format” on page 102.

If the MUST IN (=mandatory input) field attribute is assigned statically with IFG, it is ignored in lists. FHS converts the modification of field attributes by program unit (as for formats without list areas).

The new fields have the following function:

Line field

Specifies the number of list lines output. This number should never exceed the maximum number defined with IFG. Please note that a list line can occupy several lines of the screen.

Scroll info

Field with a length of four bytes with which the program unit defines the vertical and horizontal scroll commands for the next entry.
Permitted entries are the characters +, -, <, >, and blanks.

The contents of this field are displayed as scroll information on the screen; in the previous example this is “More: -”. If the field contains blanks only, FHS does not output scroll information.

MODINDEX area

This area continually lists the modified list lines. It consists of a sequence of two-byte fields with line numbers: 1 stands for line number 1, m for line number m, etc. This sequence is terminated by the number 0 (or by the end of the area, if all of the lines have been modified).

MODINDEX can be used for input and output.

MODINDEX for input:
This gives you the numbers of the list lines in which at least one input field has been modified or that have been preset as modified, e.g.

4	2	5	0	0	0

The list lines numbers 2, 4 and 5 have been modified; all other lines have remained unchanged. Please note that the line numbers are not sorted in ascending order.

MODINDEX for output: You can use this to preset a list line as modified. Fields are only evaluated up to the first 0, e.g.

3	2	6	0	0	0

The second, third and sixth list line are identified as modified in the next MGET, even if the user has not changed them.

If variable output fields to specify the displayed line range are present in the list title, the values to be output must be supplied by the application. The program unit itself must evaluate vertical and horizontal scroll information from the terminal user and must take the appropriate steps; see the following section.

Vertical and horizontal scroll commands

The terminal user can issue vertical and horizontal scroll commands. You must enter these commands in the command field (if available), or activate the function key assigned to a particular command. All the commands described on page 141 are available.

The field 'scroll info' of the previous output defines the command permitted in each situation. FHS compares the command input with the contents of the scroll info. If the command is not permitted in this situation, the program unit receives the corresponding return codes in the global attributes; see page 141.

FHS acknowledges invalid commands with an error message. In this case, control does not return to the program unit; users can correct their own input.

Valid commands are entered in the command field (if available) and transferred to the program unit.

Depending on the return code and, if appropriate, the command field, the program unit must initiate the necessary action itself, e.g. output the format with new data after it has detected a FORWARD command.

5.2.6 Commands

With FHS-DE, you can use the following command types:

– FHS commands: commands provided by FHS which are not transferred to the application (exception: vertical/horizontal scroll commands).

– application commands: commands transferred to the application.

– commands via KEY formats: commands assigned to function keys. You can assign both FHS commands and application commands to function keys. When a function key is pressed, the corresponding string is entered into the command field. A blank is inserted between the command parts. If the command field is not long enough, the command is truncated.

– combination commands: combination of direct command input and input via function keys.

If the user is to enter the command directly in a DE format, a command area must be defined in the format.

Command area

The command area of a DE format is set up when the format is created using IFG. The associated addressing aid has a field, the *command field*, where the specified commands are entered. The length of the command field is one line minus the prompt text+blank(s).

In IFG the name of the command field is freely selectable. However, two cases can be distinguished here:

- The format was created with IFG V8.0 or with IFG V8.1, where the option “Format preparation for FHS-DE compatible with IFG V8.0” was set to YES.

  When positioning, (cursor, boxes), the command field is always addressed under the name CMDAREA (unlike other input fields). If, for example, you want to position the cursor during output in the command field, you must supply the global attribute Z-CURSOR-FIELD with CMDAREA and not with the IFG name.

- The format was created with IFG V8.1, where the option “Format preparation for FHS-DE compatible with IFG V8.0” was set to NO.

  In this case, you supply the global attribute Z-CURSOR-FIELD with the IFG name.

Depending on what is entered on the screen (or, if appropriate, by the function key), FHS transfers the following data to the program unit:

– If an FHS command other than a vertical or horizontal scroll command is entered, the command field is assigned fill characters.

– Vertical and horizontal scroll commands are entered in the command field, possibly with additional operands; see page 141.

– If the input string *is not* an FHS command, this string is entered in the command field and the global attributes IKEYCL and IKEYNB are supplied with the key code.

For output formatting, the program unit can predefine a command in the command field. If this is not required, the command field should be deleted with X'00 before output. The command field can be assigned with the predefined fill character on the screen.

5.2.6.1 FHS commands

FHS commands (except for vertical and horizontal scroll commands) are not transferred to the program unit; the command field is deleted with the fill character defined when the format was created with IFG.

For some commands, the program unit nevertheless receives return codes in the global attributes. These return codes are described with the commands concerned.

FHS commands can be entered in upper or lower case.

After execution, all commands are deleted from the command field (exception: vertical/horizontal scroll commands). The command field is initialized with output fill characters. A program unit can predefine an FHS command in the command field.

Overview of the FHS commands

Operation	Meaning
ACTIONS	Place cursor in menu bar
CANCEL	Cancel display
EXIT	Exit application section
EXTHELP	Request extended help
HARDCOPY	Print contents of screen
HELP	Request help
HELPHELP	Overview of the help system
INDEX	Show index of FHS-DE keywords
KEYAREA	Activate/deactivate display of key assignment
KEYSHELP	Help on key assignment
PANELID	Activate/deactivate display of format names and message codes
SETP	Assign P keys
FORWARD BACKWARD LEFT RIGHT + - < > ++ -- << >>	Vertical and horizontal scroll commands

ACTIONS Place cursor in menu bar

Operation	Operands
ACTIONS	

ACTIONS places the cursor in the menu bar. If the cursor is already in a menu bar or in a pull-down menu, it is returned to the field in which it was positioned before the preceding ACTIONS command. If no preceding ACTIONS command was issued, the cursor is placed in the first input field of the work area.
This command can only be meaningfully used as an F key (default F10). It has no effect if no menu bar exists.

CANCEL Cancel display

Operation	Operands
CANCEL	

If CANCEL is issued for an implicit box, this box is removed without the program unit receiving a message. The underlying implicit box in the hierarchy is then displayed. If such an implicit box no longer exists, the basic format appears.

If CANCEL is issued for an explicit box or a basic format, the program unit receives the following FHS return codes in the global attributes:

MAIN RETURNCODE = 8

ERROR CATEGORY = 128

ERROR REASON = 4

The program unit should react to these return codes accordingly and go back one processing step. Control also passes to the program unit if errors are detected in fields for which an implicit validation was agreed with IFG; see page 151.

Although the input fields were checked, the output is not repeated. In the event of input errors, each field in the transfer area is marked as incorrect, i.e. the EDIT STATE field attribute has the value INVALID. The field is then filled with X’00’.

EXIT Exit application section

Operation	Operands
EXIT	

If EXIT is issued for an implicit box, all the implicit boxes are removed and the basic format is displayed. The application program is not notified.

If EXIT is issued for an explicit box or a basic format, the application program receives the following FHS return codes in the global attributes:

MAIN RETURNCODE = 8

ERROR CATEGORY = 128

ERROR REASON = 8

Although the input fields were checked, the output is not repeated. In the event of input errors, each field in the transfer area is marked as incorrect, i.e. the EDIT STATE field attribute has the value INVALID. The field is then filled with X'00'.

The application program should react to the return codes depending on the situation, and should terminate the processing step, e.g. by returning to the last synchronization point with PEND RS. Control also passes to the program unit if errors are detected in fields, for which an implicit validation was agreed with IFG; see page 151.

EXTHELP Request extended help

EXTHELP supplies general help information for the current DE format.

Operation	Operands
EXTHELP	

The help panel defined with IFG as extended help on the current format is displayed.

HARDCOPY Print contents of screen

The current screen contents are output to a hardcopy unit.

Operation	Operands
HARDCOPY	

The HARDCOPY command has the same effect as when the value HARDCOPY LOC is entered in the global attribute COPY CONTROL.

HELP Request help

This command has two formats: “Field-related help” and “Help on an FHS command”.

Field-related help

Operation	Operands
HELP	

Outputs the help text for the field where the cursor is currently positioned. This field can be an input field, a selection field, a selectable field or a command field. If a help text is defined for a protected output field, the global attribute TABULATOR CONTROL must be set to NO AUTOTAB for the format so that the cursor can be positioned in this field.

If there is no field-related help at the cursor position, general help information is displayed (global help, extended help, etc.).

It is recommended to assign the HELP command to a function key (default assignment: F1).

Help on an FHS command

Operation	Operands
HELP	command

command

FHS command for which help is requested. The cursor must be in the command field. If no FHS command is specified for ’command’, help is output on the command line. The application developer must generate this help panel with IFG.

HELPHELP Overview of the help system

HELPHELP supplies information on using the help commands.

Operation	Operands
HELPHELP	

INDEX Show index of FHS-DE keywords

The INDEX command displays a help panel containing an index of FHS-DE keywords for which help is available. See also “Help system”.

Operation	Operands
INDEX	

INDEX displays a help panel in which FHS-DE keywords are defined as cross-references. If you place the cursor on one of these keywords and press the ENTER of HELP key, help on that keyword will be displayed.

KEYAREA Activate/deactivate display of key assignment

You can activate or deactivate the display of the key assignment with KEYAREA.

Operation	Operands
KEYAREA	[ON / OFF]

ON Activate display of key assignment

OFF Deactivate display of key assignment

No operand specified

Switch display

After signing on to the UTM application, the display is set to ON. If this setting is changed during the session, the new value remains in effect until sign-off (= KDCOFF).

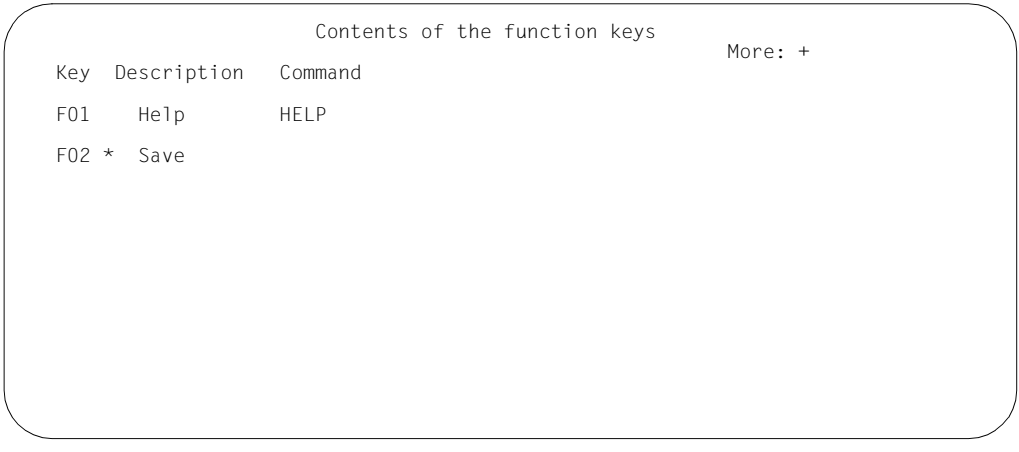
If KEYAREA ON is specified in a format that does not have an area for key display, it has no effect for the format concerned. However, key assignment display is then activated for all subsequent formats of the session.

KEYSHELP Help on key assignment

Operation	Operands
KEYSHELP	

KEYSHELP outputs the key assignment in the form of a table. This table contains the name, associated command text, and code of each key. The table occupies several screen pages. You can scroll up and down using the commands + and -.

An example of output is given below:



The F1 key requests help information and the F2 key is marked with a '*', i.e. it is assigned by UTM generation (SFUNC statement).

PANELID Display format name and message code

Using the PANELID command, you can activate or deactivate the display of format names and message codes.

Operation	Operands
PANELID	[ON / OFF]

ON Activate display

OFF Deactivate display

No operand specified

Switch display

After signing on to the UTM application, the display is set to OFF. If this setting is changed during the session, the new value remains in effect until sign-off (= KDCOFF).

SETP Assign P keys

SETP is used to manage the assignment of P keys to F keys.

Operation	Operands
SETP	Pn / (Pn, ... ,Pm) / Pn-Pm ON / OFF [, ...]

Pn	The P key with the number n.
(Pn,..,Pm)	All listed P keys Pn, .., Pm
(Pn-Pm)	All P keys from Pn through Pm
ON	The assignment is effective for the specified keys
OFF	The assignment is canceled for the specified keys

The SETP operands can be specified more than once. They must then be separated by a comma; see the following example. Each comma can have any number of blanks on either side.

Example

```
SETP P1 ON,   P2 OFF,   (P3,P7,P8) ON,   (P4-P6) OFF
```

After this SETP command, the following assignment applies:

```
P1 - F1   P3 - F3   F7 - F7   P8 - F8
```

The assignment of P2, P4, P5, and P6 is canceled.

Vertical and horizontal scroll commands

You use these commands to control vertical and horizontal scrolling in list areas. These commands can also be performed for help boxes, e.g. if the text cannot be displayed fully in a box. With help boxes, the program unit is not aware that a command has been issued.

In all other cases, the program unit is informed by return codes in the global attributes. The effect of the commands and the associated return codes are shown in the following table.

Operation	Operands	Meaning	Return code Global attribute		
			MAIN CODE	ERROR CATEGORY	ERROR REASON
FORWARD / +	[MAX / number]	Scroll forward	8	128	20
BACKWARD / -	[MAX / number]	Scroll backward	8	128	24
LEFT / <	[MAX / number]	Move to the left	8	128	28
RIGHT / >	[MAX / number]	Move to the right	8	128	32
++		Scroll to the end	8	128	20
- -		Scroll back to beginning	8	128	24
<<		Move to the left edge	8	128	28
>>		Move to the right edge	8	128	32

MAX position cursor at the end of the specified direction, e.g. +MAX = go to the end.

number number of lines, columns, or fields by which you wish to move.

The return codes are *always* transferred to the program unit (even if the format was defined without a command area). However, return codes only distinguish between four types of operation (+,-,<,>).

In order for the program unit to recognize or e.g. to distinguish '+' from '++', the format must have a command area. For formats with a command area, the *complete command* is entered together with the operands in the command field of the addressing aid. In addition to the operands listed above, the application developer can define user-specific operands and assign these to a function key.

The program unit is responsible for evaluating the return codes (as well as possibly the command field) and responding accordingly, i.e. outputting the required data (see the section "Outputting lists" on page 127ff).

After execution, the vertical and horizontal scroll commands are not deleted from the command area of the format (unlike the other commands).

RESHOW Repeat the previous screen output

This function is used to repeat the last screen output. All entries made in the meantime are lost.

This command is always linked to a K key; it cannot be entered directly in a format. By default, this function is executed by activating the K3 key; see the section “Function keys and KEY formats” on page 144ff.
You should always assign RESHOW to a K key so that you can fully restore your mask if is overwritten (by messages from asynchronous programs, for example).

5.2.6.2 Application commands

Each entry in the command field that does not represent an FHS command is transferred with the remaining entries to the program unit. In this way, you can implement “application commands”, i.e. the program unit can evaluate the input and react accordingly.

It is recommended that you evaluate the command field *before* all the other input fields, e.g. to intercept incorrect entries in the command area.

An application command is retained in the command field.

You can link an application command to a function key via a KEY format. If this key is pressed, FHS enters the associated application command in the command field. For a description of how to combine commands, see the section “Combining commands” on page 143.

5.2.6.3 Assigning commands to function keys

You can assign both FHS commands and application commands to function keys (= F keys and K keys). F keys can be simulated by P keys (SETP command).

Keys are assigned to commands via KEY formats; see the section “Function keys and KEY formats” on page 144.

F keys and K keys operate as follows:

Commands on F keys

If an F key is pressed, it has the same effect as if the command were entered in the command field and the ENTER key pressed. The command is interpreted by FHS and the input fields transferred where necessary to the program unit.

Commands on K keys

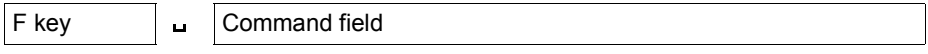
If a K key is pressed, the command is interpreted by FHS. The input fields, however, are *not* transferred to the program unit.

Note

The UTM generation takes precedence over the KEY formats, i.e. the function keys are not evaluated if they have already been assigned by means of the KDCDEF statement SFUNC.

5.2.6.4 Combining commands

If an F key is assigned a character string by a KEY format, you can combine commands by entering a character string in the command field and then activating this F key. FHS combines both character strings as follows:



FHS interprets this as *one* command. An FHS command is executed after pressing the F key. An application command is transferred to the program unit, i.e. the first MGET supplies 19Z (because of the F key); for the second MGET the command is entered in the command field of the addressing aid. If there is not enough space in the command field, the command is truncated.

Example

You want to execute the FHS command “KEYAREA OFF” as a combination command. To this end, the KEYAREA must be assigned to a key, e.g F20, in the KEY format. If you enter “OFF” in command field and then activate F20, FHS executes the required command, i.e. the F key assignment is masked out on the screen.

5.2.7 Function keys and KEY formats

You can assign commands to function keys, i.e. F keys and K keys, using FHS-DE, as well as simulate F keys with P keys.

With FHS-DE, the function keys have the following characteristics:

– If you assign a command to a function key, pressing the function key has the same effect as entering the command in the command field and pressing the ENTER key.

– When a function key that is not specified in an SFUNC statement is pressed, return code 19Z is supplied by UTM at the first MGET. A second MGET is therefore required in order to obtain the actual input. This is important if you assign the commands EXIT, CANCEL, or application commands to function keys. 19Z is not returned if the F key is simulated by a P key.
For further information on commands, refer to page 131ff.

– If a key with a command assignment is pressed and the command area is missing from a DE format, only the key code is transferred to the program unit (in the global attributes).

– If an unassigned key is pressed, control is not returned to the program unit as yet. The screen is displayed unchanged.

– The UTM generation (SFUNC) has precedence over the key assignment, i.e. a function key generated with SFUNC cannot be used in FHS-DE.

A command is assigned to a function key by means of a KEY format.

5.2.7.1 KEY formats

A KEY format (= key list) is a special format that defines a fixed assignment of function keys to commands. Some default KEY formats are supplied with FHS. Users also have the option of generating their own KEY formats with IFG (IFG format 0901) and assigning each DE format a *specific* KEY format (IFG format 0102). You can use this to restrict the effect of F keys and K keys to the current DE format only, something that is not possible by means of UTM generation (SFUNC statement).

It is, however, recommended to assign all important or frequently used function keys in the same way for the entire application; see “KEY formats of the format library” on page 145.

If a separate KEY format is not assigned to a DE format, then the key assignment of the default KEY format applies for this format; see “KEY formats of the format library” on page 145.

A KEY format contains the key assignment in the following form:

```
Fxx   ...  [command1] [text1]
Kyy   ...  [command2] [text2]
```

Here, Fxx indicates the F key with the number xx, and Kyy the K key with the number yy; command1 and command2 indicate an FHS command or an application command, and text1 and text2 indicate explanatory text. This text is output by means of the KEYSHELP command and can have a maximum of 12 characters. For further information on the command, see the section “Commands” on page 131ff.

A key can also be specified in the KEY format without a command. This is useful for keys, which, for example, are allocated by UTM generation (SFUNC statement), because it allows the FHS command KEYSHELP to display the assignment of this “SFUNC” key; see the example in the description of the KEYSHELP command.

KEY formats of the format library

For certain tasks, FHS offers a default assignment of the function keys, which are supplied in the form of the following KEY formats of the format library:

IDHKEYS	Default KEY format
IDHKEYA	KEY format for full screen with menu bar
IDHKEYE	KEY format for extended help
IDHKEYF	KEY format for field-related help
IDHKEYH	KEY format for help on help
IDHKEYI	KEY format for INDEX
IDHKEYK	KEY format for help on the keyboard
IDHKEYM	KEY format for message boxes
IDHKEYN	KEY format for message boxes without help

If a key is assigned a command, this assignment is the same for all KEY formats supplied. The following two tables show the key ←→ command assignment for the individual KEY formats in the form in which they are displayed on screen. Keys without a text entry are effective, but do not appear on the screen.

IDHKEYS - General formats

Key	Command	Designation
F1	HELP	Help
F3	EXIT	Exit
F4	HARDCOPY	
F7	BACKWARD	
F8	FORWARD	
F11	INDEX	
F12	CANCEL	Cancel
K3	RESHOW	

IDHKEYA - Key format for full screen with menu bar

Key	Command	Designation
F1	HELP	Help
F3	EXIT	Exit
F4	HARDCOPY	
F7	BACKWARD	
F8	FORWARD	
F10	ACTIONS	Menu
F11	INDEX	
F12	CANCEL	Cancel
K3	RESHOW	

IDHKEYE - Extended help

Key	**Command**	**Designation**
F1	HELP	Help
F3	EXIT	
F4	HARDCOPY	
F7	BACKWARD	
F8	FORWARD	
F9	KEYSHELP	Key
F11	INDEX	
F12	CANCEL	Cancel
K3	RESHOW	

IDHKEYF - Field-related help

Key	**Command**	**Designation**
F1	HELP	Help
F2	EXTHELP	ext. Help
F3	EXIT	
F4	HARDCOPY	
F7	BACKWARD	
F8	FORWARD	
F9	KEYSHELP	Key
F12	CANCEL	Cancel
K3	RESHOW	

IDHKEYH - Help on help

Key	Command	Designation
F3	EXIT	
F4	HARDCOPY	
F7	BACKWARD	
F8	FORWARD	
F9	KEYSHELP	Key
F11	INDEX	
F12	CANCEL	Cancel
K3	RESHOW	

IDHKEYK - Help on keyboard

Key	Command	Designation
F1	HELP	Help
F2	EXTHELP	ext. Help
F3	EXIT	
F4	HARDCOPY	
F7	BACKWARD	-
F8	FORWARD	+
F11	INDEX	
F12	CANCEL	Cancel
K3	RESHOW	Reshow

IDHKEYM - Message box with help

Key	Command	Designation
F1	HELP	Help
F3	EXIT	
F4	HARDCOPY	
F11	INDEX	
F12	CANCEL	Cancel
K3	RESHOW	

IDHKEYN - Message box without help

Key	Command	Designation
F1	HELP	
F3	EXIT	
F11	INDEX	
F12	CANCEL	Cancel
K3	RESHOW	

IDHKEYI - Index format

Key	Command	Designation
F1	HELP	Help
F3	EXIT	
F4	HARDCOPY	
F7	BACKWARD	
F8	FORWARD	
F9	KEYSHELP	Keys
F12	CANCEL	Cancel
K3	RESHOW	

Display key assignment

By default, the assignment of keys F1, F3, and F12 is displayed in the command area; when lists are output, the assignment of F7 and F8 is also displayed. You can activate and deactivate this display with the FHS command KEYAREA.

The key assignment for each DE format can be output by means of the FHS command KEYSHELP.

5.2.7.2 Simulating F keys with P keys

You can simulate F keys with P keys for the duration of a UTM session, e.g. if the data display terminal has an insufficient number of F keys. Pressing the P key has the same effect here as pressing the assigned F key.

You define the assignment of P keys/F keys by specifying the FHS command SETP during a session. SETP is described in the section “Commands” on page 131. This type of assignment is only ever valid for the duration of a session, i.e. the assignment is canceled when you sign off from the UTM application (KDCOFF).

You can modify or cancel the assignment during a session using further SETP commands (command SETP OFF). If you overwrite the contents of a P key during a session, the assignment is also canceled.

CAUTION!

If P keys intended for simulation are overwritten by means of SIDATA or a private assignment using the global attribute P-KEY-SET, undesirable side-effects may be produced as a result.

5.2.8 Validating input fields

FHS-DE offers new validation options for input fields. Validation is performed independently of the UTM application.

You define the validation options when creating the format with IFG. FHS can check input against the following stipulations:

Rule for time specifications

The input must be entered in the form HH:MM or HH:MM:SS, where HH = hours, MM = minutes and SS = seconds. The time separator, in this case the colon (:), is defined with IFG. Only one time format may be used in a mask. The colon always appears as the separator in the data transfer area.

List of values

The input must either match a value in a list of values (check for equal) or the value must not appear in the list (check for not equal). The list of values and the type of check (equal/not equal) is defined when the format is created.
Note that values entered in the value list using IFG must not have any fill characters at the start or end of the string and must not consist of only fill characters (input or output fill characters).
Example: if the fill character is a blank, it will not be possible to validate the value of a string comprising only blanks. Such a check must be implemented via the “minimum input length”; see page 31.
A value such as ‘␣PAUL’ can never return a positive result the fill character is a blank.

Range of values (numeric)

The input must be within a predefined numeric range (including minimum and maximum values). Signs are possible. For negative values, the minus sign must be specified.

Character set

The input must either consist purely of characters contained in a character list (check for equal) or the characters must not appear in the character list (check for not equal). The character list and the type of check (equal/not equal) are defined when the format is created.

If the terminal user enters invalid values, FHS informs the user of the incorrect input with a default message. The user can thus correct his or her input. This message is output in the form of a message box; see the section “Implicit boxes” on page 118. You can also define your own messages; see the section “Outputting messages” on page 153.

In two cases, control is given to the application, even if the input contains errors:

– the user has specified one of the FHS commands EXIT or CANCEL, even though there are still incorrectly filled input fields on the screen

– correction dialog is not used for the format, i.e. *NONE is specified as message ID (message code).

In these cases, the program unit must react as in FHS V7.1 or earlier, i.e. it must evaluate the global attributes and field attributes in order to detect and process the incorrect input.

For the new validation options, the field attribute group “Edit return code” has been extended to include the following return codes:

C’09’	No entry in a mandatory input field
C’11’	Value is smaller than lower limit value
C’12’	Value is greater than upper limit value
C’13’	Invalid character when checking range of values
C’14’	Value is not in list (check for equal)
C’15’	Value is in list (check for not equal)
C’17’	Character is not in character set (check for equal)
C’18’	Character is in character set (check for not equal)
C’30’	Date/time error
C’37’	Hour error
C’38’	Minute error
C’39’	Second error
C’92’	Locked selection field selected
C’93’	Selection does not exist
C’94’	Invalid character specified in multiple-choice selection field

The remaining return codes are described in the section “Field attributes” on page 62ff.

5.2.9 Outputting messages

FHS-DE can simplify the work of the terminal user by informing the user of certain events by means of DE messages. These DE messages are referred to simply as messages hereafter, and are either output in the message area of the format or in specific message boxes. FHS distinguishes between implicit and explicit messages.

Implicit messages are output by FHS independently of the application. Examples of implicit messages are messages output by FHS when checking input fields.

Explicit messages are initiated by the program unit when the global attribute MESSAGE IDENTIFICATION is supplied with a message code during format output.

Generating messages

You must generate the message formats for explicit messages yourself with IFG. To this end, call the “Process messages” mask in IFG. Define the text of the message in this mask. The message text can have a maximum length of 256 characters. In addition to the message text, you can define other characteristics:

– the message code in the form AAAAnnn, where AAAA are alphabetic characters (A-Z), and nnn are digits (0-9). The designations IDHS and IDHI should not be used as message codes, since they are reserved by FHS for internal messages.

– the output destination of the message. This can be a message area of a format, a modal message box, or a modeless message box. For further information, see the section “Implicit boxes” on page 118ff.

  If “output destination = message area” was defined, but output is not possible in the message area, FHS outputs the message in the form of a modeless box. This can happen, for example, if the message area is too small or if a message area has not been defined in the format.

– the type of message: Information/Warning/Error/Danger

– the name of a help panel for the message (optional)

You can add the characters “%%” (new line, only for boxes) and “%%%%” (blank) to the message for message output.

Editing the message text

If the string “%%” appears in the message text, the text that follows is continued in the next line of the message box as of column 2. This editing feature can be used to reduce the width of a message box. The string “%%” is replaced by a blank in the output of the message in the message area, and the string itself is removed.
“%%%%” generates a blank line.

The character “&” must be specified as “&&” in the message text.

With FHS-DE, the width of a message box is adjusted to the message text. The minimum and maximum widths for a mask are 20 and 56 characters, respectively. A message box can have up to 6 text lines. A mask width of less than 56 characters is used if the message text is shorter or if editing characters are used to set the line length. If the message text is longer, the maximum width is selected, and the message text is adjusted to it. Separation occurs at a blank.

If the edited message text exceeds the maximum size of a message box, the editing characters are replaced by blanks, and the text is split after every 56 characters. If the complete text cannot be displayed, the last line ends with “...”.
When FHD-DE encounters “&name” in the message text, it looks for a field of that name in the format and inserts the current contents of that field in the message text (assuming such a field was found). The mask field in question may have attribute “BLANKED” (invisible).

Implicit messages

Default message formats are supplied for the implicit messages. You must copy this default message format into the format library of the UTM application. Enter the name of the library with the start parameters. The default messages have the name IDHSnnn/IDHInnn, where nnn is a three-digit number.

After field validation checks, you can also have your own implicit messages output instead the default messages. The messages to be output, if any, are specified in IFG in the “Message code” field:

*NONE	No message output
Blanks	Default message of FHS
Message code	Private implicit message

User-defined implicit messages can be supplied with the following variables:

&ZPAR0	Name of the full format or partial format that defines the topmost part of the screen, or name of the active explicit box.
&ZPAR1	Name of the current field
&ZPAR2	Contents of the current field
&ZPAR3	Name of the current format

If the variable &ZPAR0, &ZPAR1, &ZPAR2 or &ZPAR3 is found in the message text, the corresponding value is substituted for it in the message text. The use of such variables is only possible for messages with message codes for which checking is defined in IFG.

These messages are output by default in a message box. You can, however, change the output location with IFG so that the message is output in the message area of the format.

Output explicit messages

You output an explicit message by supplying the global attribute MESSAGE IDENTIFICATION with the message code of the required message during format output. The message is then either output in the message area of the format or in a box, depending on what was defined when generating the message.

You can define the reference point for a message box with the global attribute MESSAGE LOCALIZATION and thus position the box on the screen. You can position the cursor at the same time by means of the global attribute DIALOG CURSOR POSITION. If the two positions conflict (cursor concealed by box), then the specifications in DIALOG CURSOR POSITION also apply for the message box, i.e. MESSAGE LOCALIZATION is ignored.

If you do not predefine a reference point for the box, FHS positions the box in the middle of the screen.

5.2.10 Help system

The application developer can create an extensive help system for the terminal user with FHS-DE. This help system can be tailored specifically for the current UTM application independently of the program unit.

Help information is displayed in the form of implicit boxes and is itself a format. You create application-specific help panels with the IFG; some default help panels are supplied with FHS. The features of implicit boxes are described on page 118.

The scrolling commands FORWARD and BACKWARD are permitted in help panels, because the size of the work area is such that all the information does not fit into one help box. For details on how to activate this help, see the section “Information for the terminal user” on page 166.

The options for application developers are listed in the following section.

5.2.10.1 Help that can be created by the application developer

The application developer can create the following help:

– extended help on the format
– field-related help on input fields, output fields and selection fields
– global help on selection fields and for the command area
– help on messages.

Extended help on the format

When you generate a format with IFG, you can allocate a help panel (which you also generate with IFG) to that format. This help is output when the user

– enters the command EXTHELP (also initiated by a function key), see the section “FHS commands” on page 132ff.

– places the cursor in a help box that was requested for field-related help and presses the “EXTHELP” key.

– places the cursor in the mask or dialog box on a field for which no field-related help exists and activates “HELP”. Note that it is only if the global attribute NO AUTOTAB is set that the cursor can be positioned anywhere on the screen. If AUTOTAB is set, the cursor can only be placed in unprotected input fields or in protected marked fields. Extended help may then need to be requested as in the first case.

– places the cursor in a field for which field-related or global help is being displayed and then presses HELP again.

If no extended help is defined but a request for it is made, a corresponding message is output.

Field-related help

In IFG mask 0306, you can assign a help panel to each field (even a protected output field).

If you define help on an output field, then it must be possible to position the cursor in this field, i.e. it must be possible either to mark the output field or the format must have the tab attribute NO AUTOTAB.

The terminal user receives the help panel if he/she positions the cursor in the field concerned and enters the HELP command, or if he/she presses the assigned F key.
If the field is part of a single-choice field, the user must enter the number of the requested field and the HELP command in the associated input field (if no number is entered, global help is provided, if available; see “Global help”).

Global help

Global help is available in the information content between the extended help which is valid for an entire format, and the field-related help which only concerns a single field. You generate global help as a help panel with IFG and assign the following objects:

– a single-choice field in IFG format 0113. The user receives this global help by positioning the cursor in the *empty* input field of the single-choice field and specifying the HELP command.

– a multiple-choice field IFG format 0114. The user receives this global help by positioning the cursor in any input field of the multiple-choice field and specifying the HELP command once or twice. If no field-related help exists, the user receives global help either immediately, or after the second HELP.

– the command area of a format (IFG format 0102). This help is activated by means of the HELP command if the cursor is in the command field and if the command field is either empty or if it contains an application command.

– the list area of a format. The user receives this help by positioning the cursor in a list field and specifying the HELP command once or twice. If no field-related help exists, the user receives the global help either immediately, or after the second HELP.
Global help is also output if the cursor is placed in the list area outside a field and HELP is pressed.

Help is output in a box. For output fields, this box should be below the input field concerned if possible, and for the command area, it should always appear above the command field.

Help on messages

If you want to define a help text for a message, specify the required help format explicitly when generating the message format.

This help panel is output if the terminal user presses the HELP key after the message is output. For message output in a box, the cursor must always be located within the message box.

Cross-references

Cross-references are hypertext links to text fields in help panels. They are similar to field-related help in action formats. The user can obtain information on a cross-reference by placing the cursor in a text field for which help is defined and pressing the ENTER key or issuing a HELP command. This causes a new help box to be output with help information displayed in it. Further cross-references may be contained in this box.

If a hierarchy of cross-references is requested, the corresponding help boxes are always output at the same position. Consequently, only the topmost box will be visible if all cross-references are of the same size. This should be taken into account when designing the help panel.

INDEX

The INDEX command allows you to search for help on FHS-DE. On entering the INDEX command, you receive a help panel containing keywords for FHS-DE. If you place the cursor on any keyword and press the ENTER key, a new help box is output with help information displayed in it. This box may contain new cross-references.

5.2.10.2 Help offered by FHS

A range of default help panels is supplied with FHS. The names of these help panels begin with IDHH. These help panels must be copied into the format library of the application before starting the application.

FHS offers the following help to the terminal user:

– help on messages from FHS
– help on FHS commands
– help on key assignments
– help on the help system
– help on FHS-DE (index)

Help on messages

The help on messages from FHS is handled like the help on the application-specific messages; see above.

Help on FHS commands

The user can request help on a certain FHS command either by entering HELP 'fhs-command' in the command field and pressing the ENTER key, or by entering the command and pressing the HELP key. The help on FHS commands appears as global help. If application-specific help exists for an FHS command, the user receives the application-specific help with the first HELP call, and the FHS help for the second.

Help on key assignment (key list)

The help on key assignment can be activated by two methods:

– The terminal user issues the FHS command KEYSHELP, or presses the assigned F key.

– The cursor is placed in the bottom frame of the box in the key list display and the HELP command is issued.

FHS displays the assignments for all F keys and K keys in a table; see the description of the KEYSHELP command. Only those keys for which entries were made when creating the KEY format are shown. If all key assignments are not visible, you can scroll forward or backward as required.

When the key list is displayed, extended help may be requested:

– If a help panel was defined when assigning keys, the EXTHELP command displays that help panel. If the cursor is outside the command area, the same help can be displayed with the HELP command.

– If no help panel was defined, the EXTHELP command displays the message “No help exists”. If the cursor is outside the command area, the HELP command displays help on the KEYSHELP command.

The associated default help panel ((IDHKHLP) can be modified to a certain extent in IFG, e.g. by modifying text fields in titles, display attributes, or the explanatory text for ’*’, which is enclosed within < >. This text is located in the command field in the first data line of IDHKHLP.

You cannot, however, change the structure of this format. If the structure of the IDHKHLP format is destroyed by more radical changes, the KEYSHELP function can no longer be implemented.

Help on the help system

The help on the help system supplies information on the help available and how it can be used. The FHS command HELPHELP or the corresponding F key activates this help.

Help on FHS-DE

This help is displayed via the INDEX command or from cross-references of some standard help texts. It contains information on working with FHS-DE.

5.2.11 Cursor handling in the program

The program unit can determine the position of the cursor at input and can position the cursor at output using the global attribute DIALOG CURSOR POSITION; see page 102ff. The following rules apply:

Cursor position at input

If the cursor is in a designated field at input, the global attributes of the attribute group DIALOG CURSOR POSITION supply the position of the cursor, where Z-CURSOR-FIELD contains the field name.

If the cursor is located in a text field or between designated fields, the following values are entered in the global attributes:

Z-CURSOR-FIELD:	absolute position of the cursor in the form $lll#ccc
Z-CURSOR-POSITION:	0
Z-CURSOR-INDEX:	0

Cursor position at output

The specification in Z-CURSOR-FIELD determines where a DE format is output. The following rules apply here:

1. If Z-CURSOR-FIELD contains an absolute position ($lll#ccc), the specifications in Z-CURSOR-POSITION and Z-CURSOR-INDEX are ignored.
2. If Z-CURSOR-FIELD contains a field name, Z-CURSOR-POSITION specifies the move in the field. The following also applies:
   - If the field name refers to a list line, the line number must be specified in Z-CURSOR- INDEX.
   - If the field is a normal field that appears only once on the screen, the value 0 must be specified in Z-CURSOR-INDEX.
   - If the field appears under the same name in several partial formats, the rank of this field must be specified in Z-CURSOR-INDEX.
   - If the value in Z-CURSOR INDEX or the field name in Z-CURSOR-FIELD is invalid, a formatting error occurs. To combat this, FHS corrects an incorrect value in Z-CURSOR-POSITION and sets it to 0.
3. If the attributes Z-CURSOR-FIELD, Z-CURSOR-INDEX, and Z-CURSOR-POSITION are not supplied, the rules for FHS V7.1, or later apply in the case of a basic format. The cursor is positioned in the first input field for a box.

5.2.12 Language extensions

FHS-DE offers you the option of processing the following dialog elements depending on the language used:

– default format
– user-defined formats
– help panels
– message formats
– KEY forrmats

FHS makes the distinction on the basis of a language extension. The language extension must be a capital letter (’A’ to ’Z’). The eighth character of a format name under which the format is stored in the format library is used as a language extension.

You can use this, e.g. to create formats for several languages and store these in such a way that the only difference between format names is the eighth character. The application can use the required format depending on the language used.

There are two options for assigning language extensions:

1. in the global attribute LANGUAGE EXTENSION for each MPUT call
2. in the default format IDHSLNG (applies to the entire application)

You can use one, both, or none of these options. If you wish to use both options, the specification for the MPUT call takes precedence over the specification in IDHSLNG.

In addition to language extensions, you can also define language-specific prompts for scrolling commands (default format IDHSCRL).

Details on the above options are given in the following sections.

Language extension specified in the MPUT call

If you want to work with a format-specific language extension, enter the required language extension in the global attribute LANGUAGE EXTENSION of an MPUT call. FHS appends the language extension to the format name and searches for the format under the complete name.

If the format name in KCMF is less than 7 characters, FHS extends it to seven characters with “#” characters.

Example

```
KCMF = #FORMAT3
LANGUAGE EXTENSION = D     Format name in format library: FORMAT3D

KCMF = #FORMAT3
LANGUAGE EXTENSION = E     Format name in format library: FORMAT3E

KCMF = #FORM1
LANGUAGE EXTENSION = F     Format name in format library: FORM1##F
```

If FHS does not find the format under the complete name, it searches under the original name (entry in KCMF).

Example

A format is output with the language extension for “French”. A default DE message, e.g. for a value check, has the message number IDHS023. In this case, the message member “IDHS000F” would be searched first for the message (if “F” is the extension for French), and if no such member were found, the member “IDHS000” would be searched next.

The language extension is valid for all implicit dialog elements connected with the format specified by MPUT. Help, message, and KEY formats are implicit dialog elements.

Default language extension in IDHSLNG

You can define a language extension for implicit dialog elements using IDHSLNG, i.e. for help, message, and KEY formats linked with the current MPUT call. This extension is used if you have not specified a language extension with MPUT.

To define the default, call IFG and enter the required language extension in IDHSLNG in the top left-hand-hand corner. During supply, IDHSLNG contains a blank. Before starting the application, copy IDHSLNG into the format library.

You should define “D” as the language extension for “German” and “E” as the language extension for “English”, since FHS uses both of these language extensions for all supplied formats. If you wish to use some other language, you should follow the same conventions as for the default formats supplied.

Default prompt in IDHSCRL

You can define your own language-specific prompts for scrolling information in help panels. To this end, you must create a format with the name IDHSCRL and copy it into the format library before starting the application. The following example shows the structure of this format:

```
D:Weiter:
F:A Suivre
I:...
S:...
*:More:
```

There must be a language extension before the first colon (= 8th character of the language-specific help panel). Text after the colon can be up to 11 characters in length. A maximum of 5 specifications in different languages is permitted. You can define the last entry as the default value with ’*’; in the example, the English text is the default. This default is used if FHS cannot allocate the language extension.

If no default is defined, the last entry is taken as the default.

As of IFG V8.1, prompts for help panels are defined in the help panels themselves. The setting in IDHSCRL is not used in such cases.

Working without language extensions

If you want to work without language extensions, you must observe three rules:

1. The IDHSLNG format must not contain uppercase letters (A-Z). Enter an '*' (asterisk) in IDHSLING as the extension.
2. For the MPUT call, you must supply the global attribute LANGUAGE EXTENSION with blanks or binary zero. FHS then searches for the format under the name you have specified in KCMF (as was the case for FHS V7.1 or earlier).
3. All the default formats supplied with FHS must be copied into the format library in such a way that the last character of the format name is omitted, e.g. the help panel IDHKHLPD becomes help panel IDHKHLP.

5.2.13 Information for the terminal user

The terminal user can receive messages from FHS-DE and can activate FHS-DE functions. Messages are either output in the message area of the format (bottom line) or in a separate message box; see the corresponding section on page 118f.

You activate the FHS-DE functions by entering the FHS command in the command field or by pressing the allocated F key. The F key assignment for some default cases is displayed in the command area. The structure of a DE format and the meaning of the individual areas are described on page 100.

The following is an overview of the FHS commands.

Operation	Meaning
ACTIONS	Place cursor in menu bar
CANCEL	Cancel display
EXIT	Exit application section
EXTHELP	Request extended help
HARDCOPY	Print contents of scree
HELP	Request help
HELPHELP	Overview of the help system
INDEX	Display index of FHS-DE keywords
KEYAREA	Activate/deactivate display of key assignment
KEYSHELP	Help on key assignment
PANELID	Activate/deactivate display of format names and message codes
SETP	Assign P keys
FORWARD BACKWARD LEFT RIGHT + - < > ++ -- << >>	Vertical and horizontal scroll command

The commands and their effects are described starting on page 131.

Explanation of the commands

- You enter a command either by pressing the assigned F key or by writing the command in the command field and pressing the ENTER key. You can also combine both options; see the section “Combining commands” on page 143.
  If boxes appear on the screen, the cursor position is also significant.
- ACTIONS places the cursor in the menu bar; see also page 132.
- EXIT and CANCEL refer to processing. The effect of these commands depends on both the situation and the application; see the description of CANCEL and EXIT, page 131ff.
- Except for messages, HELP delivers help to the field where the cursor is currently located. Depending on the type of field, the following applies:
  - if the cursor is in a single-choice field with a valid entry, help on this entry is supplied.
  - if the cursor is in an empty single-choice field, help is output on the entire single-choice field (= global help).
  - if the cursor is in the input field of a multiple-choice field, help is output on this input field. If the HELP command is issued again, global help is output on the multiple-choice field (if available), otherwise extended help is output.
  - if the cursor is in the command field and if this is empty or if it contains an application command, global help is output on the command field. If this is not available, FHS outputs default help text on the command area.
    If the command field contains an FHS command, the help is output to this command.
  - if there is no help on a normal input field, extended help is output on the format. If this is not available either, FHS outputs a corresponding message.
- EXTHELP supplies extended help on the last format output by the application with MPUT. Any help already on the screen is removed beforehand.
- You receive help on a message by pressing the F key assigned to the HELP command, irrespective of where the cursor is located. If help is displayed, the message disappears.
- Help texts can be too long to fit into a box. If such is the case, you can enter scroll commands (+,-,...) to obtain more information. The box then contains explanatory text, e.g. “More: +”.

5.2.14 Information on using FHS-DE

You can only use the functionality of FHS-DE with UTM V3.3 or later. As before, you structure program units using INIT, MGET, MPUT, and PEND; see the openUTM manual “Programming Applications with KDCS for COBOL, C and C++”.

Some functions, however, may also be used for existing UTM applications (openUTM V3.3 or later) without having to change the program units. The next section describes how to achieve this.

If you wish to use all the new functions of FHS V8.3, you must perform the following steps:

1. Delete all default formats of former versions of FHS (IDH*) from your format library.
2. Copy all default formats of SYSFHS.FHS.083 to your format library.

FHS-DE functions for existing applications

The following DE functions can be used without modifying the program unit:

– help panels for field-related help and help panels for an entire format
– key assignment for a format
– checks on input fields

To use these functions, you must convert the existing #formats (IFG /FHS V7.1 or earlier) such that these formats can use implicit boxes. Proceed as follows:

1. Call the IFG function “Modify format” and activate the “Dialog processing” option.
2. Extend the format by the corresponding functions; possible functions are:
   – help panels for field-related help on input fields
   – define key assignment of the format
   – help panel for the entire format (=extended help)
   – extended checks for input fields
3. Generate the associated help panels, message formats, and KEY formats with IFG and store these in the format library.

Restrictions and special features

The following restrictions and special features apply when using DE formats:

- Update formatting is not supported for DE formats.
- FPUT/DPUT
  - It is not possible to print DE formats (FPUT/DPUT).
  - DE formats cannot be output asynchronously to terminals using FPUT
- Merging DE formats with previous FHS formats
  - Do not merge DE partial formats and normal FHS partial formats to form an entire format.
  - A box cannot be output in a normal FHS format.
  - A normal FHS format cannot be output in a box.
- DE formats cannot be preformatted with IFG.
- The nesting depth of boxes is restricted by the size of the PI.

  The size of the save area for DE formats is 64K. The following is contained in it:

For each full format, partial format, or explicit box:	1 x data transfer area (addressing aid)
For each screen level	1 x control information for screen level: approx. 100 bytes 1 x combined data transfer area: combination of individual data transfer areas based on dynamic format 1 x dynamic format (created from combination of formats involved. The length roughly corresponds to that of the DE format. You can determine this length by loading the format and viewing it with AID)
For implicit formats	1 x second dynamic format (combination of dynamic format and help/messages)

Note

If you wish to work using help, your DE format and the associated data transfer area must not exceed 16K. Each field in your mask requires at least 60 bytes in the format.

- Partial formats
  - A DE partial format cannot be output in boxes.
  - DE partial formats are processed in the order in which they appear on the screen, whereas normal FHS partial formats are processed in the order in which MPUT calls were issued. Global attributes may therefore be evaluated differently. To minimize these differences, FHS-DE takes the global attribute of the first DE partial format, for which the relevant attribute is not X'00'.

    Global attributes of partial formats that are not output again, are not evaluated (exception: Start Line). In the case of a DE application, all partial formats that are displayed at the same time on the screen must be compatible with one another. This means that all format types must be the same in the following points:
    - They must have the same format type (e.g. action format, help format)
    - They must have the same number of columns
    - They must have been created for the same screen size
    - They must be “colored” or “non-colored” (same color table)
    - They must have the same background color
    - They must follow the same global editing guidelines
    - 8-bit formats must be defined for the same terminal group in the case of 8-bit formats

    This means, for example that if a special partial format has some fields with a color attribute then the field attribute COLOUR must be defined for the fields of all partial formats displayed on the same screen.
  - A screen may comprise a maximum of 20 partial formats.
- Global attribute UNDEFINED

  This attribute can be set to 'Y' for DE formats, even if no fields are undefined. The reason for this may be that an undefined field can become a significant field during intermediate dialog, and FHS-DE cannot carry out any more checks after intermediate dialogs.
- P keys after restart
  - The P keys are loaded during restart for DE formats if the global attribute LEVEL SELECTION has the value LEVEL-P. For normal FHS formats, P keys are not loaded even if LEVEL-P is set.
  - Simulation of F keys by P keys is lost after a restart, i.e. the user must issue the SETP command again.

- F keys
  - FHS-DE can only use those F keys that were not generated in a SFUNC statement for UTM.
  - If the terminal user presses an F key to enter an FHS command that causes the application to return (CANCEL/EXIT), or an application command, the first MGET in the program unit supplies the return code 19Z. The data must then be read with a second MGET.
- Conversation stacking

  Conversation stacking is also possible with DE formats. Since several formats can appear on a screen, however, the inserted program unit must carefully check which formats are currently on the screen when an MPUT PM call is issued, so that a formatting error does not occur.

  Partial formats can be exchanged with MPUT PM. However, if this occurs, entries made before stacking will be lost.
- KDCFHS is possible for DE formats, but with INIF and INFD it only supplies the format specifications of the last box or last full format displayed.

  The service function KDCFHS can be called with operation codes ’INIF’ and ’INIL’ to reset the field attributes and some global attributes of a data transfer area of a DE format. ’INIF’ and ’INIL’ return the same result. Operation code ’INIL’ must be used if a format with a language extension is to be initialized.

  The language extension must be entered in the “LANGUAGE” field in the COPY element “FHSCUSER” for COBOL programs. The “LANGUAGE” field is not evaluated for operation code ’INIF’.

  The name(s) of the format(s) that was/were formatted at the current screen level is/are returned by the service with operation code ’INFD’. In other words, if a box was output last, the function returns the name of that box, and if a full format was output, the name of the full format or the names of all partial formats is/are returned.

  The language extension, if present, is ignored.
- Pseudo format #!POPUP

  With MPUT, a box is output with the format name #!POPUP; if necessary, you can issue a second MPUT for the actual format name. The following INIT supplies the name of the format that belongs to the uppermost box remaining (or to the basic format), but not #!POPUP.

  If, however, a formatting error has occurred in the meantime, #!POPUP is supplied by the following INIT call in the KB program area. UTM aborts the conversation after the MGET.

- Transaction processing and restart for DE formats
  - openUTM saves formats and explicit boxes, i.e. formats output by the application. Message codes transferred by global attributes are also saved.
  - Implicit boxes and other actions inserted by FHS are not saved.
  - If, after a restart, work is continued on another terminal, the other terminal must also belong to the terminal group defined in the format. If several DE formats appear on the screen, the uppermost active box determines the terminal group. For partial formats, this is the uppermost partial format on the screen. Error information can also be displayed in an error mask if the output of a box is not possible. Errors detected during implicit actions result in the termination of the implicit action, but do not usually abort the transaction. The application can be continued normally. The terminal group is defined in IFG and can be modified using IFG.
- Errors when formatting DE formats

  In some cases, if errors occur when formatting DE formats, which cause the conversation to abort, FHS-DE outputs a message box containing error information beforehand. In such cases, e.g. if the data transfer area contains incorrect entries, the conversation is aborted by the following input (F, K, or ENTER key) containing a formatting error.

  Thus it is easier to evaluate errors in the test phase of the application.
- Remove incorrect data

  If incorrect data is entered and if FHS does not rectify the error, this data is removed under the following conditions:
  - after input of the PANELID or KEYAREA command
  - after UTM has requested a screen refresh, e.g. after asynchronous intermediate messages
- FHS modules in common memory pool

  If FHS modules are assigned to a common memory pool, the UTM application must be regenerated for FHS V8.3, since the MFHSEUAS module was integrated in the prelinked module MFHSROUT as of FHS V8.0.

5.2.15 POPUP-CB data structures

The data structures for the POPUP control block are in the SYSLIB.FHS.083.xxx library. Apart from the Assembler structure, the members are all of type S (Assembler: type M).

Assembler data structure IASPOPUP

```
POPUPCB  DS    0F
POPUPRP  DS    C
POPUPAP  DS    C
         DS    H
POPUPRL  DS    F
POPUPAL  DS    F
POPUPAC  DS    F
POPUPAN  DS    CL8
         MEND
```

C header file ICCPOPUP.H

```
typedef struct {
    char    rempop;
    char    addpop;
    char    filler[2];
    long    rm_lev;
    long    ap_line;
    long    ap_col;
    char    ap_name[8];
} popup_cb;
```

COBOL data structure IDHPOPUP

```
      35 POPUP-CB.
                  41  POPUP-REMPOP   PIC X(1).
                  41  POPUP-ADDPOP   PIC X(1).
                  41  FILLER         PIC X(2).
                  41  POPUP-RM-LEV   PIC S9(5) COMP.
                  41  POPUP-AP-LINE  PIC S9(5) COMP.
                  41  POPUP-AP-COL   PIC S9(5) COMP.
                  41  POPUP-AP-NAME  PIC X(8).
```

DRIVE data structure IDRPOPUP

```
        19 POPUP_CB,
          20 POPUP_REMPOP            CHAR(1),
          20 POPUP_ADDPOP            CHAR(1),
          20 FILLER                  CHAR(2),
          20 POPUP_RM_LEV            INTEGER,
          20 POPUP_AP_LINE           INTEGER,
          20 POPUP_AP_COL            INTEGER,
          20 POPUP_AP_NAME           CHAR(8);
```

Fortran header file IFOPOPUP

```
      CHARACTER  * 24  POPUPCB
*
*
      CHARACTER  *  1  POPUPREMPOP
      CHARACTER  *  1  POPUPADDPOP
      INTEGER    *  4  POPUPRMLEV
      INTEGER    *  4  POPUPAPLINE
      INTEGER    *  4  POPUPAPCOL
      CHARACTER  *  8  POPUPAPNAME
*
*
      EQUIVALENCE (POPUPCB ( 1: 1),  POPUPREMPOP)
      EQUIVALENCE (POPUPCB ( 2: 2),  POPUPADDPOP)
      EQUIVALENCE (POPUPCB ( 5: 8),  POPUPRMLEV)
      EQUIVALENCE (POPUPCB ( 9:12),  POPUPAPLINE)
      EQUIVALENCE (POPUPCB (13:16),  POPUPAPCOL)
      EQUIVALENCE (POPUPCB (17:24),  POPUPAPNAME)
```

Pascal data structures

IPAPOPUP-BODY data structure

```
PACKAGE BODY O;
BEGIN
END.
```

IPAPOPUP-SPEC data structure

```
PACKAGE POPUP;
(* POPUP CONTROL BLOCK *)

TYPE T_POPUP_CB           =
  RECORD
    POPUP_REMPOP           (00000) :  CHAR;
    POPUP_ADDPOP           (00001) :  CHAR;
    FILLER                 (00002) :  PACKED ARRAY
                                           (.01..02.) OF CHAR;
    POPUP_RM_LEV           (00004) :  INTEGER;
    POPUP_AP_LINE          (00008) :  INTEGER;
    POPUP_AP_COL           (00012) :  INTEGER;
    POPUP_AP_NAME          (00016) :  PACKED ARRAY
                                           (.01..08.) OF CHAR;
  END;
```

PL/I data structure IP1POPUP

```
DECLARE
01 IDHPOPUP,
       19 POPUP_CB,
         20  POPUP_REMPOP          CHAR,
         20  POPUP_ADDPOP          CHAR,
         20  FILLER                CHAR(2),
         20  POPUP_RM_LEV          BINARY FIXED(31),
         20  POPUP_AP_LINE         BINARY FIXED(31),
         20  POPUP_AP_COL          BINARY FIXED(31),
         20  POPUP_AP_NAME         CHAR(8);
```

RPG data structures

IRPPOPUPI data structure

```
I*  POPUP CONTROL BLOCK
I                                       1   1 PPUPRP
I*                                                    POPUP REMPOP
I                                       2   2 PPUPAP
I*                                                    POPUP ADDPOP
I                                     B 5   8OPPUPRL
I*                                                    POPUP RM LEV
I                                     B 9  12OPPUPAL
I*                                                    POPUP AP LINE
I                                     B 13 16OPPUPAC
I*                                                    POPUP AP COL
I                                       17  24 PPUPAN
I*                                                     POPUP AP NAME
```

IRPPOPUPO data structure

```
O*  POPUP CONTROL BLOCK
O                        PPUPRP     1
O*                                       POPUP REMPOP
O                        PPUPAP     2
O*                                       POPUP ADDPOP
O                        PPUPRL     8B
O*                                       POPUP RM LEV
O                        PPUPAL    12B
O*                                       POPUP AP LINE
O                        PPUPAC    16B
O*                                       POPUP AP LINE
O                        PPUPAN    24
O*                                        POPUP AP NAME
```

5.2.16 Example of dialog extension

This section contains general instructions on how to convert the functions of the dialog extension to a program unit. An example in COBOL is given below together with detailed explanations.

Program structure for job control

The following structure diagram shows the essential parts of job control.

UTM application program unit	
	INIT operation
	MGET operation depending on KCRMF
	Initialization of data transfer area depending on program state
<--	Evaluation of return code for EXIT and CANCEL
<--	Command processing
<--	Check on input fields
	Processing routines
	MPUT operation depending on processing state
	PEND operation

If the basic format and the dialog box(es) are processed by the same program unit, then after the INIT, the program unit must read the KCRMF field to check which format was output in the previous cycle. Depending on KCRMF, the MGET call that suits the previous format is executed.

Commands should always be evaluated *before* the other input fields. Application commands can, for example, be used for certain “services” of the application or for quick job control in accordance with “expert mode”. During format generation with IFG, you should assign the attribute “Conversion to uppercase” to the command field.

The FHS-DE test options should be used to check the field contents.

Information on dialog boxes

When editing dialogs that process data from boxes, you must observe the following with respect to the data transfer area:

If the data transfer areas of the basic format and the dialog box are located in the SPAB (standard primary working area), as recommended by UTM, the program receives only the data of the active format with MGET, i.e. either from the basic format (or partial formats), or from the active dialog box.

If, with the next MPUT, data entered in a dialog box is to be displayed in the underlying basic format, you must refill the data transfer area of the basic format completely, i.e. the program must save previous data itself, e.g. in the communication area (KB) or in a LSSB (local secondary storage area). A REMPOP without a format MPUT would only display the previous mask without modifications.

You can use this method to implement a “prompt” for format fields. You can, for example, select the prompt character as an “undefined value” (see IFG, “Editing attributes”). If this character is entered (e.g. ’?’ as in the following example), you save the contents of the other data fields in the KB or in an LSSB and output the dialog box, possibly with data, read from a database. The selected data from the box is processed during the next cycle and is entered in the prompt field of the data transfer area instead of the prompt character (if the entries were correct). Finally, this is completed with the saved data. After a REMPOP, output the basic format in the same program unit with MPUT.

You can also use LSSBs for list processing.

Programming example in COBOL

Unicode formats are supported only with COBOL 2000.

This section contains the most important extracts from a UTM-COBOL program unit. You can easily add the extensions necessary for an executable program yourself, if you are familiar with the programming rules for UTM programs.

The sample program shows a simple dialog which consists of viewing and removing a box by activating one of the defined keys. The program performs the following steps:

– The DE full format (#DEFORM) is output on the terminal. It contains a command area with the command field CMDAREA and a KEY format.

– The return code for the commands EXIT and CANCEL is evaluated.

– The entry “DATE” in the command field is accepted as an application command. After ’DATE’, the program outputs the current date in a box; the day of the week is displayed in the relevant language. The date is taken from the KB header.

Program statements are not necessary to apply the dialog functions ’help panels’ and ’standard checks on fields’ to the example. The application command DATE can also be activated using a function key, if it is contained in the KEY format.

The most important points of the sample program are explained in the conclusion; the corresponding reference numbers are in the comment lines.

The DE format is mapped to three dialog situations below: a date box, a help box, and a prompt dialog box. The contents of the boxes are shaded for clarity.

```
                              Project management
--------------------------------------------------------------------------------

Please select an action: Z +

Project no.:        : .......... +
Project name        : ............................... +
Project type        : . +

Autor               : ........................
Version             : .......
Programming system  : ............

Begin date          : ........
Current status      : .

--------------------------------------------------------------------------------
COMMAND: ==> DATE
F1=HELP    F2= ---------------------------------------------- 12=cancel
               |    Today is Friday the 16th of Oct. 1992    |
--------------------------------------------------------------------------------
```

DE format #DEFORM with an explicit dialog box for displaying the date

```
                              Project management
--------------------------------------------------------------------------------

Please select an action: Z +
                                        ----------------------------
Project no.:        : .......... +     |    Help on action        |
Project name        : .............    | ------------------------ |
Project type        : . +              |   E - Enter              |
                                       |   M - Modify             |
Autor               : .............    |   Z - Show               |
Version             : .......          ----------------------------
Programming system  : ............

Begin date          : ........
Current status      : .

--------------------------------------------------------------------------------
COMMAND: ==> DATE
F1=HELP    F2= =FKEY display  F3=Exit  F5=Prompt  F11=Techni  F12=Cancel

--------------------------------------------------------------------------------
```

DE format #DEFORM with implicit help box

```
                              Project management
--------------------------------------------------------------------------------

Please select an action: Z +

Project no.:        : .......... +
Project name        : ..................................  +
Project type        : ? +
                       ----------------------------------------
Autor               :  | Please select a project type:         |
Version             :  | _ 1. Purchasing                       |
Programming system  :  |   2. Sales                            |
                       |   3. Inventory management             |
Begin date          :  |   4. Materials management             |
Current status      :  |                                       |
                       | F1=Hilfe  F3=Ende                     |
                       ----------------------------------------
--------------------------------------------------------------------------------
COMMAND: ==> DATE
F1=HELP    F2= =FKEY display  F3=Exit  F5=Prompt  F11=Techni  F12=Cancel

--------------------------------------------------------------------------------
```

DE format #DEFORM with explicit prompt dialog box

COBOL source program

```
          .
          .
          .
 WORKING-STORAGE SECTION.
*
*   UTM operation codes, FHS attribute values
*
     COPY KCOPC   SUPPRESS.
     COPY FHSAVAL SUPPRESS.
 LINKAGE SECTION.
*
*   Communication area (KB)
*
     COPY KCKBC SUPPRESS.
       05  KBPROG PIC X(100).
     03   IDHX-PROG-KB REDEFINES KCKBPRG.
     COPY IDHXKBC.
          .                                    Program KB              1)
          .
          .
       40 KBXMASK.
         41 KBXMASK1                  PIC X(08).
         41 KBXBOX1                   PIC X(08).

       40 KBXLANG                     PIC X.
       40 KBXSTATUS                   PIC X.
          88 INIT-MAP                 VALUE "M".
          88 F-KEY                    VALUE "F".
          88 K-KEY                    VALUE "K".
          88 E-KEY                    VALUE "I".
          88 BOX1                     VALUE "B".
          88 SERV-TAC                 VALUE "S".
          88 EXIT-TAC                 VALUE "X".
          88 CLEAR-DATA               VALUE "C".

       40 FHS-RETURN-CODES.
*
        42 GA-RC-MAIN                 PIC 9(5) COMP.
          88 MC-OK                    VALUE 0.
          88 MC-WARNING               VALUE 8.
        42 GA-RC-CATEGORY              PIC 9(4) COMP.
          88 CATEGORY128              VALUE 128.
        42 GA-RC-REASON                PIC 9(4) COMP.
          88 REASON-CLEAR             VALUE 0.
          88 REASON-CANCEL            VALUE 4.
          88 REASON-EXIT              VALUE 8.
```

```
          88 REASON-FORWARD            VALUE 20.
          88 REASON-BACKWARD           VALUE 24.
          88 REASON-LEFT               VALUE 28.
          88 REASON-RIGHT              VALUE 32.

       40 KBXMESSAGE.
         41 KBXMSSID                   PIC X(04).
         41 KBXMSSNR                   PIC X(04).

       40 KBXDAY-NO                    PIC 9(04) COMP.
          .
          .
          .
*
*      SPAB (only valid from INIT through PEND)
*
     COPY KCPAC SUPPRESS.
*
*      POPUP  - control block
*
   03  ASSGNB      PIC S9(18) SYNC.
   03  POPUP-CB-NB.
     COPY IDHPOPUP.
*
*      Format - addressing aid (for dialog box)
*
   03  ASSGN3      PIC S9(18) SYNC.
   03  DATE NB.
     COPY DATE.
      41 MAPDATE-R REDEFINES MAPDATE.
        42 MAPDATE-YEAR   PIC 99.
        42 MAPDATE-DEL1   PIC X.
        42 MAPDATE-MONTH  PIC 99.
        42 MAPDATE-DEL2   PIC X.
        42 MAPDATE-DAY    PIC 99.
*
*      Format - addressing aid (for full mask)
*
   03  ASSGN       PIC S9(18) SYNC.
   03  DEFORM-NB.
     COPY DEFORM.
*
      41 COMM5-TAB  REDEFINES CMDAREA OCCURS 2 TIMES.
        42 COMM5X     PIC X(05).
*****************************************************************
```

```
 PROCEDURE DIVISION USING KCKBC KCSPAB.
 FLOW SECTION.
 ABL-01.
     PERFORM   INIT-OPERATION.
     IF KCRCCC NOT = ZERO
        THEN   MOVE INIT TO F-OP
               PERFORM ERROR-BEH
               GO TO ABL-99.
*                                                      MGET:                2)
     MOVE MGET  TO KCOP.
     MOVE KCRMF TO KCMF.
     IF KCRMF  =  "#DEFORM"
       THEN        PERFORM MGET-MASK
       ELSE
       IF KCRMF  = "#DATUM"
        THEN       PERFORM DATUM-PROC
        ELSE
*                                          if no MGET issued
                   PERFORM MGET-ZERO.
*
     IF KCRCCC > "01Z"
        THEN   MOVE MGET TO F-OP
               PERFORM ERROR-BEH
               GO TO ABL-99.
*
     IF INIT-MAP
        THEN   PERFORM MASK-INIT.
*
*    Terminate program if EXIT or remove data if CANCEL
*                                                                           3)
     IF MC-WARNING
        THEN  IF REASON-EXIT
                 THEN   SET EXIT-TAC TO TRUE
                        PERFORM PEND-FI-OPERATION
                        GO TO ABL-99
                 ELSE   IF REASON-CANCEL
                        THEN   SET CLEAR-DATA TO TRUE
                               PERFORM MASK-INIT.
```

```
*
*    Handling the other F keys and P keys
*
     IF F-KEY
        THEN   PERFORM FKEY-PROC
        ELSE
*
*    Main processing section (after SEND - key)
*
     IF E-KEY
        THEN   PERFORM SEND-PROC.
*
*    Output section and PEND
*
     IF BOX1
        THEN   PERFORM MPUT-BOX-OPERATION
        ELSE   PERFORM MPUT-OPERATION.
     IF KCRCCC NOT = ZERO
        THEN   MOVE MPUT TO F-OP
               PERFORM ERROR-BEH
               GO TO ABL-99.
     PERFORM   PEND-OPERATION.
 ABL-99.
     EXIT  PROGRAM.
*****************************************************************
      .
      .
      .
*************************
 MGET-ZERO SECTION.
*************************
 MGET-1.
*
     SET  INIT-MAP   TO TRUE.
     MOVE "D"        TO KBXLANG.
     MOVE "#DEFORM"  TO KBXMASK1.
     MOVE SPACE      TO KBXBOX1.
*
     MOVE MGET     TO KCOP.
     MOVE KCRLM    TO KCLA.
     MOVE KCRMF    TO KCMF.
     MOVE SPACES   TO KCRN.
     CALL "KDCS"  USING KCPAC NB.
     IF KCRCCC  = "19Z"
       THEN
         MOVE KCRMF    TO KCMF.
         MOVE SPACES   TO KCRN.
         CALL "KDCS"  USING KCPAC NB.
```

```
          .
          .
          .
*  Initial values for FHS return codes
     SET MC-OK        TO TRUE.
     SET REASON-CLEAR TO TRUE.
 MGET-9.
     EXIT.

*************************
 MGET-MASK SECTION.
*************************
 MGET-1M.
*
*   Format name #DEFORM is already in KCMF.
*
     MOVE MGET       TO KCOP.
     MOVE 556        TO KCLA.
     MOVE SPACES     TO KCRN.
     CALL "KDCS" USING KCPAC DEFORM-NB.
     IF KCRCCC  = "19Z"
        THEN
*                             second MGET supplies data
        CALL "KDCS" USING KCPAC DEFORM-NB.
     MOVE RC-MAIN    OF DEFORM-GLOBALS TO GA-RC-MAIN.
     MOVE RC-REASON  OF DEFORM-GLOBALS TO GA-RC-REASON.
     MOVE INPUT-KEY-CLASS OF DEFORM-GLOBALS TO KBXSTATUS.
*
 MGET-9M.
     EXIT.

*************************
 MASK-INIT SECTION.
*************************
 MINIT-1.
*
* Initialization of global attributes       ---DEFORM---
     MOVE SPACES    TO DEFORM-NB.
     MOVE ZERO      TO CURSOR-POS     OF DEFORM-GLOBALS.
     MOVE ZERO      TO STARTLINE      OF DEFORM-GLOBALS.
       ...
* Predefine data fields if appropriate
*
 MINIT-9.
     EXIT.
```

```
**************************                                          4)
 SEND-PROC SECTION.
**************************
 SEND-1.
*
     IF CMDAREA NOT = SPACES
        THEN          PERFORM COMM-PROC
                      GO TO SEND-9.
*
     IF FIELDS-UNDEFINED OF LITEST-GLOBALS = GA-UNDEFINED
        THEN          PERFORM PROMPT-CHECK-PROC
                      GO TO SEND-9.
*
*  If the field ACTION was changed, then ...
*
     IF (INPUT-STATE-ACT OF ACTION-FAB = FA-MODIFIED)
        THEN ...

     END-IF.
*
       .
       .
       .
 SEND-9.
     EXIT.

**************************                                          5)
 COMM-PROC SECTION.
**************************
 COMM-1.
*
     IF ( COMM5X (1) = "DATE" )
        THEN    MOVE "#DATE"  TO KBXBOX1
                MOVE SPACE     TO DATE-NB
                MOVE ZERO      TO CURSOR-POS OF DATE-GLOBALS
                MOVE ZERO      TO STARTLINE  OF DATE-GLOBALS
                PERFORM        TODAY-PROC
                MOVE LOW-VALUE TO POPUP-CB
                MOVE "CMDAREA" TO POPUP-AP-NAME OF POPUP-CB
                PERFORM        INSTALL-BOX
                SET BOX1       TO TRUE
                GO TO COMM-7.
*
*  Unknown application command:                                     6)
*
     MOVE "222"     TO KBXMSSNR.
     MOVE "CMDAREA" TO MSG-LOC       OF DEFORM-GLOBALS.
     GO TO COMM-8.
```

```
 COMM-7.
     MOVE SPACES TO CMDAREA.
 COMM-8.
     MOVE ZERO   TO Z-CURSOR-POS   OF DEFORM-GLOBALS.
     MOVE ZERO   TO Z-CURSOR-INDEX OF DEFORM-GLOBALS.
     IF (KBXMSSNR NOT = SPACE)
       THEN
             MOVE "CMDAREA" TO Z-CURSOR-FIELD OF DEFORM-GLOBALS
           ELSE
             MOVE "ACTION"  TO Z-CURSOR-FIELD OF DEFORM-GLOBALS
       END-IF
 COMM-9.
     EXIT.

*************************                                          7)
 FKEY-PROC SECTION.
*************************
*
 FKEY-1.
*
     IF INPUT-KEY-CLASS OF LITEST-GLOBALS = GA-F-KEY
        THEN
             PERFORM COMM-PROC.
 FKEY-9.
     EXIT.

*************************                                          8)
 TODAY-PROC SECTION.
*************************
 TODAY-1.
*
     MOVE KCTJHVG     TO KBXDAY-NO.
     ADD 2            TO KBXDAY-NO.
     DIVIDE KBXDAY-NO BY 7 GIVING WORK2 REMAINDER WORK1.
     IF WORK1 = 0
        THEN          MOVE 7 TO WORK1.
*  Sets the day of the week for the specific language
     IF KBXLANG = "D"
        THEN          MOVE WTAG(WORK1) TO MAPWDAY
        ELSE          MOVE WDAY(WORK1) TO MAPWDAY.
     MOVE "-"         TO MAPDATE-DEL1
     MOVE "-"         TO MAPDATE-DEL2
     MOVE KCTAGVG     TO MAPDATE-DAY
     MOVE KCMONVG     TO MAPDATE-MONTH
     MOVE KCJHRVG     TO MAPDATE-YEAR
*
 TODAY-9.
     EXIT.

*****************************************************************
*  Sections for box functions ADDPOP and REMPOP                 *
*****************************************************************
```

```
*************************                                          9)
 INSTALL-BOX SECTION.
*************************
 INST-1.
     MOVE "A"    TO POPUP-ADDPOP.
     MOVE SPACE  TO POPUP-REMPOP.
*
     PERFORM MPUT-POPCB.
*
 INST-9.
     EXIT.
*************************
 MPUT-POPCB SECTION.
*************************
 MPOP-1.
     MOVE MPUT       TO KCOP.
     MOVE "NT"       TO KCOM.
     MOVE "#!POPUP"  TO KCMF.
     MOVE 24         TO KCLM.
     MOVE ZEROES     TO KCDF.
     CALL "KDCS" USING KCPAC POPUP-CB-NB.
 MPOP-9.
     EXIT.
*************************
 REUSE-POPCB SECTION.
*************************
 REUSE-1.
*
     MOVE SPACE   TO POPUP-ADDPOP.
     MOVE SPACE   TO POPUP-REMPOP.
*
     PERFORM MPUT-POPCB.
*
 REUSE-9.
     EXIT.
*************************
 REMPOP-1 SECTION.
*************************
 REMP-1.
*
     MOVE SPACE    TO POPUP-ADDPOP.
     MOVE "R"      TO POPUP-REMPOP.
     MOVE 1        TO POPUP-RM-LEV.
*
     PERFORM MPUT-POPCB.
*
 REMP-9.
     EXIT.
```

```
*************************
 REMPOP-ALL SECTION.
*************************
 REMPA-1.
*
     MOVE SPACE    TO POPUP-ADDPOP.
     MOVE "R"      TO POPUP-REMPOP.
     MOVE 0        TO POPUP-RM-LEV.
*
     PERFORM MPUT-POPCB.
*
 REMPA-9.
     EXIT.
*************************                                          10)
 DATE-PROC SECTION.
*************************
 MGET-1D.
*
* "#DATE" is already in KCMF
     MOVE KCRLM      TO KCLA.
     MOVE SPACES     TO KCRN.
     CALL "KDCS" USING KCPAC DATUM-NB.
     IF KCRCCC = "19Z"
        THEN
*                             second MGET supplies the data
        CALL "KDCS" USING KCPAC DATE-NB.
*
*  Always terminate the function (for all keys)
     SET E-KEY TO TRUE.
     PERFORM REMPOP-1.
     PERFORM PEND-OPERATION.
*
 MGET-9D.
     EXIT.
*************************                                          11)
 MPUT-OPERATION SECTION.
*************************
 MPUT-1.
*
     IF KBXMSSNR NOT = SPACES
        THEN MOVE KBXMESSAGE TO MSG-IDENT  OF DEFORM-GLOBALS.
     MOVE KBXLANG    TO LANGUAGE-EXT       OF DEFORM-GLOBALS.
     MOVE MPUT       TO KCOP.
     MOVE "NE"       TO KCOM.
     MOVE "#DEFORM"  TO KCMF.
     MOVE 556        TO KCLM.
     MOVE ZEROES     TO KCDF.
```

```
     MOVE SPACE      TO KCRN.
     CALL "KDCS" USING KCPAC DEFORM-NB.
*
 MPUT-9M.
     EXIT.
**************************                                          12)
 MPUT-BOX-OPERATION SECTION.
**************************
 MPUT-1B.
*
     MOVE KBXBOX1    TO KCMF.
     MOVE MPUT       TO KCOP.
     MOVE "NE"       TO KCOM.
     MOVE ZEROES     TO KCDF.
     MOVE SPACE      TO KCRN.
     MOVE KBXLANG    TO LANGUAGE-EXT OF DATE-GLOBALS.
     MOVE 118        TO KCLM.
     CALL "KDCS" USING KCPAC DATE-NB.
*
 MPUT-9B.
     EXIT.
```

Explanations

1. Some condition names are defined in the program communication area, which reflect the current dialog and program state. They are used for program job control.
2. To carry out the correct MGET operation, a check is run to see if the format #DEFORM, the (box) format #DATE, no format, or an unknown format was output on the terminal during the previous cycle.

   From the return code 19Z following MGET, the program unit can determine that messages were sent following activation of a function key. A second MGET is required to read the data. Correspondingly, the condition name is set for the KBXSTATUS program state in the MGET-MASK section when the global attribute is being transferred.
3. After activation of the EXIT key (or after input of the EXIT command plus SEND key), a return code is transferred to the global attributes by FHS-DE. The same applies for the CANCEL command. These return codes are evaluated here. After EXIT, the dialog step in the example is terminated with PEND FI; after CANCEL the format #DEFORM is output without the data it contained previously having been processed. The dialog is continued.
4. The SEND-PROC section is the actual processing procedure. It is first checked whether an application command was entered in the CMDAREA field (= command field). If so, this is processed in the COMM-PROC section. The other input fields of the format are

then checked and processed; this is not carried out in the example. If necessary, a “prompt” is also implemented for selected fields, e.g. by outputting a dialog box with a selection field.

5. CMDAREA is evaluated in the COMM-PROC section. Only the DATE command is accepted; an error message is output for other commands (see 6.).

   The output of the dialog box is edited with the DATE command, i.e. an ADDPOP is executed in the INSTALL-BOX section using the command field as a reference point (field name CMDAREA). This initializes the data transfer area for the format #DATE. Since the move values POPUP-AP-LINE and POPUP-AP-COL are both zero, FHS can move the box so that it can always be displayed.

6. The error message with the code MESS222 should be output with the format if a command not recognized by the application was entered. The message number is first stored in the KB field KBXMSSNR. The message is to be output in a message box, the reference point for the box being the CMDAREA field. The MPUT-OPERATION section (see 11.) contains the rest of the steps for outputting this message in a specific language. In the event of error, the cursor should be in the CMDAREA field, or otherwise in the ACTION field.

7. Since the #DEFORM format has a command area, the application commands activated by F keys can be evaluated in the COMM-PROC section.

8. This section edits the current date from the UTM-KB header where the day of the week is output in the relevant language. The routine shown here is only valid for the year 1992.

9. These are the sections for installing, reusing, and removing boxes.

10. The DATE-PROC section implements the MGET call for the dialog box (like MGET for the full format). All keys (SEND, EXIT, CANCEL) are handled similarly and lead to removal of the box (REMPOP-1). Since only one box was output, the unchanged basic mask is displayed by means of the PEND issued immediately without format MPUT; the command DATE is still in the command field. To avoid this, the program must be extended as described in the section “Example of dialog extension” on page 177f), by replacing, for example, the PEND-OPERATION routine with a RESTORE-DEFORM-DATA routine, and by extending the COMM-PROC section to include a SAVE-DEFORM-DATA procedure.

11. This section implements the output of the full format #DEFORM in a specific language with the relevant key list. If KBXMSSNR contains a number, the associated message is output in the relevant language, and the message code is transferred to the global attribute MSG-IDENT.

12. MPUT call for the output of the format in the box previously installed.

5.3 Service functions

This section describes the service functions you can use with the KDCFHS or KDCSCUR calls.

5.3.1 KDCFHS

The FHS service functions are called with **CALL “KDCFHS” USING USER-PARAMS area**. They can be called at any time between the KDCS calls “INIT” and “PEND”. The USER-PARAMS data structure is made available to COBOL users as COPY element “FHSCUSER”. It has the following structure:

```
*************************************************************
*  FHSCUSER Version 810                                     *
*  Copy Element for transfer parameter                      *
*************************************************************
*
 40 FILLER                   PIC S9(5) COMP SYNC.
*
 40  USER-PARAMS.
*
      41 USER-CODE           PIC X(04).
*
      41 MAP-NAME            PIC X(08).
*
      41 MAP          REDEFINES MAP-NAME.
*
         42 CONTROL-CHAR     PIC X.
*
         42 NAME             PIC X(07).
*
      41 AREA-LEN            PIC 99999 COMP.
*
      41 RETURN-LEN          PIC 99999 COMP.
*
      41 ERROR-CODE          PIC X(04).
*
      41 LANGUAGE            PIC X(01).
*
      41 RESERVED-01         PIC X(03).
```

Description of the data fields

USER-CODE

The service function must be entered in this field. The following entries are possible:

INIF
Initializing the data transfer area with separate attribute blocks and field contents (only for #formats). All the field attributes are supplied with their default values in the format. The field contents and global attributes remain unchanged (except for the global attributes for formatting acknowledgment). Data transfer areas that have already been supplied with data can thus be reset to their initial status at any time.

INIL
Returns the same result as INIF. Operation code ´INIL´ must be used if a format with a language extension is to be initialized. The language extension must be entered in the “LANGUAGE” field in the COPY element “FHSCUSER” for COBOL programs.

INFD
Supplies the number and names of formats currently on the screen.

MAP-NAME

Contains the format identifier (format handling character and format name), up to 8 characters (only evaluated with ’INIF’).

AREA-LEN

Specifies the length of the ’area’ made available by the user.

RETURN-LEN

The length provided for the required information is entered in this field. The contents of ’area’ are only valid in this length.

ERROR-CODE

An error code for the program unit, in printable characters, is entered in this field if errors occur or warnings are issued when processing the function, e.g. if the area is too small to contain all the information. The entries in this field have the following meaning:

0000	No error occurred.
ER01	The USER-CODE syntax is incorrect.
ER02	The AREA-LEN specification is too small (INFD call).

ER03 No formats have been entered (INFD call).

ER04 No DE format was specified when calling the INIL function or the format structure is invalid.

All other error codes correspond to the additional FHS return codes (see page 513ff).

LANGUAGE

Language extension for DE formats (see also page 171).

area

When the 'INIF' function is used, the data transfer area of the #format whose field attributes are to be reset must be specified here.

When the 'INFD' function is used, the area which is to contain the format names must be specified here. 'area' has the following structure:

```
40 INFD-PARAMS.
   41 USED-ENTRIES        PIC 9999 COMP.
   41 MAP-TABLE OCCURS 43 TIMES.
      42 MAP-NAME         PIC X(08).
      42 MAP      REDEFINES MAP-NAME.
         43 CONTROL-CHAR        PIC X.
         43 NAME                PIC X(07).
```

This structure is made available to COBOL users as COPY element "FHSCINFD". The format names supplied are only valid up to the index USED-ENTRIES (number of entries) and correspond to the formats currently on the screen when the call is made.

5.3.2 KDCSCUR

You position the cursor for *formats or +formats with the subprogram KDCSCUR. The address of the required field is given as a parameter to KDCSCUR. Depending on the FHS start parameters selected, this can be a data field or an attribute field (only with "+" formats). For further information see the description of the FHS start parameter "CURSOR=", in the section "Start parameters" on page 196.

5.4 Loading the formatting program

The connection modules (FHSCON, FHSCON2, IDHDHS, IDHDOOR) used for formatting are loaded from the library specified with the KDCDEF control statement 'FORMSYS' (description of the formatting system) (see also the openUTM manual "Generating Applications"). FORMSYS is specified as follows:

```
FORMSYS TYPE=FHS,LIB=libname
```

If nothing was specified for LIB=, an attempt is made to load the connection modules from the system file TASKLIB.

Connection module FHSCON is not shareable. It can be linked statically to the application program:

```
//INCLUDE-MODULE E=(FHSCON),T=R,LIB=libname
```

FHSCON2, IDHDHS, and IDHDOOR can also be loaded as shareable modules, either in the class 4 memory using a system administrator command or in the common memory pool when UTM is generated (see the openUTM manual "Generating Applications").

The formatting program modules are loaded from the library assigned with the SET-FILE command

```
/SET-FILE-LINK LINK=MROUTLIB,FILE-NAME=libraryname
```

If no such allocation has been made, the FHS modules are loaded from the library specified in the FORMSYS statement or from the TASKLIB. The FHS modules, with the exception of MFHSISYS, can also be loaded as shareable modules.

The format application file which contains the formats of the application (including the default formats of FHS-DE) is assigned with the following UTM start parameter:

```
.FHS MAPLIB=libname
```

The default is F.MAPLIB.

The FHS-DE default formats are supplied with the SYSFHS.FHS.083.FHS-DE library and must be copied to the format application files before use. If necessary, the language extension must be modified beforehand; see the section "Language extensions" on page 162 for further information.

If you wish to work with the product FHS-DOORS as of V2.0, you must replace the pre-linked module IDHDOOR in the library of FHS modules by the IDHDOOR module supplied with FHS-DOORS. If IDHDOOR was already linked into the application, you will need to relink the new module.

5.5 Start parameters

Start parameters can be assigned for an application for formatting purposes. These define application-specific default values which deviate from the FHS defaults, and also further formatting attributes not contained in the data transfer area. The start parameters are prefixed with ".FHS". The FHS start parameters are specified like the start parameters in openUTM.

Overview of the FHS start parameters

Start parameter		**Evaluated for**		
Prefix	**Operands**	**#format**	***format**	**+format**
.FHS	ALLATTR=NO/YES			x
	ATAB=YES/NO		x	x
	CURSOR=ATTR/NOATTR		x	x
	DE=YES/NO	x	x	x
	EFFLEN=YES/NO/FLDLEN		x	x
	ERASE=RSET/RSON		x	x
	EXIT=exitroutine/(exitroutine,library)	x		
	ISTD=RMOD/RUNP		x	x
	KCRLM=MEAL/MUIL		x	x
	MAPCNT=number	x	x	x
	MAPDET=(char3,char4)		x	x
	MAPLIB=library	x	x	x
	MEMLEN=length-1	x	x	x
	NILS=YES/NO		x	x
	PADDING=([FORM*='␣'/OUTMSG/char1]		x	
	[,FORM=OUTMSG/char2])			x
	PMOD=NO/YES		x	x
	RESFORM= format1/(format1[,format2.,formatn])	x	x	x
	UPDATE=ONLY/PSTN		x	x

5.5.1 Start parameters for all format types

.FHS DE=YES/NO

DE= Defines the FHS functionality.

YES The dialog extension of FHS is to be made available.

NO The dialog extension is not to be activated. FHS is therefore only available in the functionality of FHS V7.1, i.e. no DE formats can be displayed. FHS-DOORS Version 2 cannot be called with the new functions.

.FHS MEMLEN=length-1

MEMLEN=length-1

Length of the restart area which contains, among other things, the data required for restarting FHS. This length must be specified in Kbytes. The value for 'length-1' may be between 1 and 64, the default being 64.

Standard values for the size of the restart area:

for full formats

Length of the longest addressing aid + 100 bytes.

for partial formats

Sum of the lengths of all addressing aids for partial formats that are displayed on the screen at the same time + the length of the longest of these addressing aids + 2200 bytes.
The applicable value is the format configuration on screen with the greatest total length that can occur within an application.

DE formats

If FHS-DE is used, the value 64 KB is adopted here (regardless of the MEMLEN input).

.FHS MAPLIB=library

MAPLIB=library

Name of the format application file (in accordance with BS2000 conventions) which contains the formats used. If this parameter is not specified, the F.MAPLIB library is used.

.FHS MAPCNT=number

MAPCNT=number

Number of format entries in the directory. A value from 0 to 2730 can be specified for 'number'. MAPCNT=100 is the default. If the number of formats to be loaded is greater than the specified value, MAPCNT is adjusted automatically.

.FHS RESFORM=format1/(format1[,format2..,formatn])

RESFORM= This parameter is used to notify FHS of the formats that are to be loaded when an application is started and not just when they are required. The maximum number of characters that can be entered here is 4094. These specifications are additive in effect.

Note

– Only formats required frequently should be specified here. This enhances the performance of the UTM applications, but does slow down the start procedure.

– Message formats and KEY formats can also be specified here.

– National language-specific formats *cannot* be specified here.

5.5.2 Start parameters for #formats

.FHS EXIT=exitroutine/(exitroutine,library)

EXIT= This parameter is used to make the exit routine available. During formatting, the exit routine is executed for the fields for which it is requested. If only the exit routine name (up to 8 characters) is specified, the routine is loaded from the library from which FHS was loaded (see page 17). If 'library' is also specified, the exit routine is loaded from the specified library.

5.5.3 Start parameters for *formats and +formats

.FHS PADDING=([FORM*='␣'/OUTMSG/char1][,FORM=OUTMSG/char2)**.**

PADDING= Controls how the data transfer area is handled by UTM in the length of the addressing aids before the input message is entered.

FORM*= Parameter for *formats

FORM= Parameter for +formats

'␣' The data transfer area is overwritten with blanks before the input message is entered (default for *formats).

OUTMSG The data transfer area is filled with the data transfer area of the last output before the input message is entered (default for +formats).

Note
In the case of selectable fields, the fill characters only apply for the current input.

char1/char2 The data transfer area is filled with the specified character before the input message is entered. Entries may be made in the form C'x' or X'xx'.

.FHS UPDATE=ONLY/PSTN

UPDATE= Defines how variable format data is to be output if no change of format takes place.

ONLY Only data fields which do not contain X'00' in the data transfer area are output on the screen. Fields which only contain X'00' are retained.

PSTN Only fields accessible to the program are output on the screen. Fields containing X'00' are filled with fill characters.

.FHS ERASE=RSET/RSON

ERASE= Defines how the contents of the format will be erased, if this is required.

RSET The specified format will be "reset".

RSON Has the same effect as RSET and ONLY combined, i.e. new fields can be output and the remaining unprotected fields are deleted.

.FHS ALLATTR=NO/YES

ALLATTR= Controls evaluation of the attribute fields in the data transfer area for output formatting if UPDATE=ONLY or ERASE=RSON is set; evaluated for +formats only.

NO Only attribute fields in which the associated data fields do not contain X'00' are evaluated.

YES Every attribute field is evaluated, i.e. under some circumstances less data is transferred if the field attributes are to be altered on the screen.

.FHS MAPDET=(char3,char4)

MAPDET= Defines how the fields that can be selected by the user are to be transferred to the application program.

char3 Character with which selectable fields that have not been selected are filled. The default is X'00'.

char4 Character with which selectable fields that have been selected are filled. The default is X'FF'.

Notes

– char3 and char4 must be enclosed between single quotes, e.g. FHS MAPDET=('-','+').

– If the letter B is specified for char3 and char4, the blank character (X'40') is used as the fill character.

.FHS ISTD=RMOD/RUNP

ISTD= Specifies the read mode for input.

RMOD Only the modified fields are transferred.

RUNP All unprotected fields are transferred.

Note
If #formats are to be formatted with other format types in a cycle (mixed mode), ISTD=RMOD and NILS=YES must be specified.

.FHS EFFLEN=YES/NO/FLDLEN

EFFLEN= Specifies the values that are to be entered in the data transfer area's length fields when input formatting takes place.

YES The length fields contain the effective length of the associated field.

NO The effective length is not determined; the contents of the length fields remain unchanged.

FLDLEN If the field was modified, the defined length of the field is entered.

.FHS PMOD=NO/YES

PMOD= Applies only for output formatting and if the parameter UPDATE=ONLY is specified.

NO Fields modified or marked when the last input was made are set to not modified or not marked.

YES Fields modified or marked when the last input was made retain this status.

.FHS NILS=YES/NO

NILS= Controls how NULL characters are handled for input formatting.

YES Reading with NULL characters; must be specified in the event of mixed mode with #formats.

NO Reading without NULL characters. If NULL characters occur in the field and it has been defined as unaligned, the remaining characters are left-justified.

.FHS KCRLM=MEAL/MUIL

KCRLM= Specifies which length field FHS enters in return field KCRLM when MGET is used.

MEAL The length of a format's addressing aid is supplied (total length of the addressing aid).

MUIL The data length actually entered is supplied, up to the last field entered or to the end marker.

Note

If KCRLM=MUIL, the contents of the addressing aid from the last field entered or from the end mark to the end of the addressing aid remain undefined (incl. PADDING).

.FHSCURSOR=ATTR/NOATTR

CURSOR= Defines what effect the call “CALL KDCSCUR” has.

ATTR The cursor is positioned on the start of the first data field with the attribute ’cursor’ (only for +formats).

NOATTR The cursor is controlled via the address specified with KDCSCUR. This allows the cursor to be positioned anywhere in fields that are accessible to the program.

Notes

– Only CURSOR=ATTR is permissible for partial formats.
With full formats, only the last KDCSCUR call is effective.

– With CURSOR=ATTR, the cursor can also be controlled with the “CALL KDCSCUR” call when #formats are being used provided these formats use the field attribute group ’Attribute Combination’.

.FHS ATAB=YES/NO

ATAB=

YES Automatic tabulator. The cursor jumps from the end of an unprotected or markable field to the start of the next unprotected or markable field.

NO No automatic tabulator. Terminal users must position the cursor themselves.

5.6 Messages

The messages from FHS to the UTM application which are described here are sent to the logical system file SYSOUT by openUTM when an application is started. In an application that is running, these messages are sent to the associated terminal.

The messages and the formats are supplied in German and English. The sources generated by IFG (LMS type F), the modules prepared for use (LMS type R), and the profiles (LMS type U) are contained in the library.

The following messages are output in German if “D” is specified as the language extension in the default format IDHSLNG. Otherwise, they appear in English.

Message structure:

Message header	Message data; dependent on the message number

1 **4** **5**

Structure of the message header:

Bytes	Format	Meaning
1 - 2	C' FC'	Identifier for an FHS message
3 - 4	C' nn'	Message number

Messages

Message header	Message	Meaning / Response
FC01	FHS V08.3Axx bereit. FHS V08.3Axx ready.	V08.3Axx is the current version number.
FC02	Version der Schnittstelle UTM-FHSCON ist falsch Invalid version of interface UTM-FHSCON	Check versions of UTM and FHS; if required, change to correct versions.
FC03	Fuer die Schnittstelle UTM-FHSCON ist die Zugriffsmethode falsch Invalid access-method for interface UTM-FHSCON	The access method used is invalid. Check versions of UTM and FHS; if required, change to correct versions.
FC04	Fehler bei der Verarbeitung im FHS: zzzzzzzz/iixx/yyyy Error while processing FHSCON: zzzzzzzz/iixx/yyyy	An error occurred while processing in FHSCON where: zzzzzzzz: is the FHSCON return code; see table below. value '0000000A' when the module FHSCON2 cannot be loaded. ii: internal code, pinpoints the error. xx: analyzes the code yyyy; see below. yyyy: additional return code; see below.
FC05	Syntax-Fehler in der Startparameter-Karte Syntactical error in startup direction	There was a syntax error in the start parameter statement. Check the syntax and alter it if necessary.
FC06	Fehler in der Startparameter-Auspraegung Semantical error in startup direction	Semantic error in the start parameter statement, e.g. invalid value specified. Check and alter if necessary.
FC07	xx/yyyy Fehler erkannt vom FHS-Kern xx/yyyy error detected by FHS	iixx/yyyy designates a return code and an additional FHS return code. For the meaning, refer to page 513ff.

Message header	Message	Meaning / Response
FC08	Formatsteuerzeichen passt nicht zum IFG-Typ Map-control-character does not match with IFG-type	A format ID (*, + #) does not match the IFG type; Compare format ID in KDCS call and format type in IFG and adjust as required.
FC09	CALL “KDCSCUR” ist in diesem Fall nicht erlaubt In this state CALL “KDCSCUR” is not permitted	The KDCSCUR call is not permitted here, e.g. call for #format, partial format and CURSOR=NOATTR
FC10	Mischen von FHS- und FHS-DE-Formaten ist nicht erlaubt. Mix of FHS- and FHS-DE-formats is not permitted	FHS formats and FHS-DE formats must not be combined in one mask.
FC11	Interner Fehler waehrend der Verarbeitung in FHS: iixx/yyyy Internal error while processing FHS: iixx/yyyy	An internal error was found during FHS operation; the code specified corresponds to that of FC04; see tables below. The format should be altered.
FC12	Fehler waehrend der Dialogverarbeitung im FHS: iixx/yyyy Error while dialog processing of FHS: iixx/yyyy	An error occurred during dialog processing of FHS; the code specified corresponds to that of FC04; see tables below.
FC13	FHS-DE-Formate sind in dieser Umgebung nicht erlaubt FHS-DE-formats are not permitted in this environment	FHS-DE formats are to be output. However, the dialog extension is not available since FHS was called with a UTM version <3.3.

Message header	Message	Meaning / Response
FC14	Der Modul IDHDHS kann in der FHS-Bibliothek nicht gefunden werden Module IDHDHS not found in FHS library	The module IDHDHS is not in the library from which FHS is to be loaded.
FC15	Das Format kann in der Format-Bibliothek nicht gefunden werden Format not found in format library	The required format could not be found in the format library.
FC17	Falsches Format fuer Vor-einstellungen Wrong format for initial values	The structure of a format from which default values are to be obtained for initializing FHS cannot be processed or has an invalid content (IDHSLNG, IDHSCRL,..).
FC18	POPUP-Steuerblock ist falsch oder nicht im ersten MPUT POPUP control block is wrong or not in first MPUT	Error in MPUT with the POPUP control block; possible causes: Incorrect entries in POPUP control block, e.g. incorrect characters for ADDPOP or REMPOP, or the number of REMPOPs is too big. An MPUT NT was already specified before the MPUT with the POPUP control block.
FC19	Teilformate sind in einer Box nicht erlaubt Partial format in box not permitted	A format defined as a partial format is to be output in a box. However, partial formats are not permitted in a box.
FC20	MFHSCTAB kann in der FHS-Bibliothek nicht ge-funden werden Module MFHSCTAB not found in FHS library	The module MFHSCTAB is not in the library from which FHS is to be loaded. (MFHSCTAB = code tables)
FC21	Fehler in MPUT-Parame-tern oder im Datenbereich Error in MPUT parameter or data transfer area	Error in MPUT call; possible causes: In an FHS-DE format, the message format ID does not begin with ' #' . The data transfer area was not correctly supplied with global attributes, field attributes and field contents.

Message header	Message	Meaning / Response
FC22	Formate mit verschiedenen CCS-Namen kombiniert Formats with different CCS-names are mixed	An image is to be made up of formats in which the different CCS names were agreed.
FC23	CCS-Name im Format und BS2000 kleiner V10 CCS-name in format and BS2000 version lower than V10	A CCS name was agreed in this format, but a BS2000 version < V10 is used, i.e. XHCS is not supported.
FC24	Nach POPUP wurde kein Format angegeben, und es existiert keine Box No format after POPUP-CBL and a box does not exist	No MPUT with format name after MPUT with POPUP-CB and ADDPOP.
FC25	Nicht kompatible Teile des FHS werden verwendet Incompatible moduls of FHS are used	An FHS prelinked module is used that belongs to another version of FHS. Modules from another FHS library may have been loaded in a common memory pool.
FC26	Gemeinsame Verwendung von Teil- und Vollformaten Mix of partial and fullformats is not permitted.	Partial formats and full formats must not be mixed in a mask.
FC27	Fehler bei der Fehlerbehandlung, erster Fehlercode: iixx/yyyy Error during error processing origin error code: iixx/yyyy	An error occurred during dialog processing for which a message was to be output. An error alsooccurred during message output, which cannot be dealt with and therefore causes an abort. iixx/yyyy designates the return code and the additional return code of the original error; seeFC04. The meaning of ii, xx and yyyy is given in the tables below. Possible causes: A format contains so many fields that internal areas overflow when outputting implicit boxes. Response: Simplify format. The KEY format for the message box was not found in the format library.

Message header	Message	Meaning / Response
FC30	Die Version von DOORS ist unvertraeglich mit FHS Version of DOORS is not compatible with FHS	The DOORS initialization reports that the versions of FHS and DOORS are incompatible. FHS does not carry out editing for DOORS. The FHS outputs to the FE terminal can only be output in the simulation window. The message provides information during task initialization and does not abort the application.
FC31	Fehler waehrend der Verarbeitung im DOORS: iixx/yyyy Error while processing with DOORS: iixx/yyyy	An error occurred in the DOORS processing modules; ii, xx and yyyy are the return codes as for FC04; they are described in the tables below.
FC32	Keine Aktion durch FHS, da Verarbeitung am Front-End-Terminal erfolgt No DE actions due to front end processing	The front end processing reported that all actions of the FHS dialog extension have been carried out, e.g. help or checks. The FHS dialog extension, however, recognized on the basis of the data returned, that not all actions have been carried out. FHS-DE does not perform these actions either. This message is a warning of which the application is not notified. Input fields can be marked as incorrect if there is no error handling facility. Possible causes: Messages or KEY formats have different processing states on the BS2000 system and on the front end terminal.
FC33	Die Ausgabeformatierung ist nicht moeglich, da schon ein Fehler anliegt. Output formatting isn't possible; a formatting error has been occured	A previous formatting operation was terminated with an error. The formatting should be repeated using conversation stacking.
FC34	Das Format konnte nicht geladen werden oder es hat einen falschen Typ Format not found or wrong use of this format	A format that was preformatted for a printer on a data display terminal. A format cannot be found in the format library (see BLS load message) or the format is not of the desired type. Example: a help panel was requested, but the format was generated in IFG as a key list format.
FC35	Arabische DE-Teilformate haben verschiedene Modi Arabic DE-formats with different global mode	Partial DE formats for Arabic are to be displayed in a screen, but the partial formats do not all have the same global mode.

Message header	Message	Meaning / Response
FC36	Die Eingabe von der Datensichtstation kann nicht verarbeitet werden Input message from terminal is wrong	The input message from the terminal cannot be analyzed by FHS. Several read attempts were made without success. Check with your system administrator; the terminal is probably not supported.
FC37	Unicode-Teilformat in Nicht-Unicode-Formatierungszyklus Unicode partial format in not-Unicode formatting cycle	If a Unicode partial format must be output, then the first partial format of the cycle must be Unicode
FC38	Vorformatiertes Teilformat in Unicode-Formatierungszyklus Preformatted partial format in Unicode formatting cycle	a preformatted partial format may not be output in a Unicode formatting cycle
FC39	Unicode-Fromat und OSD-Version < V6.0B Unicode format and OSD version < V6.0B	Unicode is only supported from OSD V6.0B

Error codes in the FCnn messages

The following error codes can occur in the FCnn messages:

zzzzzzzz	FHSCON return code (only for FC04)
ii	internal information, not valid for the application
xx	analysis for yyyy
yyyy	additional return code

The exact meaning is listed in the following tables.

zzzzzzzz	Meaning
00000064	Combination of FHS-DE formats and */+formats not permitted
00000065	POPUP control block was not output with first MPUT
00000066	The format cannot be found
00000067	Mix of FHS partial format and FHS-DE partial format
00000068	Mix of FHS-DE partial format and FHS partial format
00000069	Error in PI management (internal error)
0000006A	Error in ADDPOP/REMPOP
0000006B	Mix of different CCS names in one mask
0000006C	The IDHDHS module cannot be loaded
0000006D	Not enough main memory
0000006E	Incorrect structure of a format for defaults
0000006F	Dialog extension not possible with FHS-DE since an openUTM version < V3.3 was used
00000070	The MFHSCTAB module cannot be found in the FHS module library
00000071	Partial formats are not permitted in a dialog box
00000072	Several full formats are to be output on a screen
00000073	Mix of full formats and partial formats
00000074	Internal error (error when positioning in the PI)
00000075	Blank message area for MPUT
00000076	Internal error (invalid format specification in the FHS save area)
00000077	The current operating system does not support XHCS.
00000078	The version of the module IDHDHS differs from that of FHSCON.
00000079	The version of the module FHSCON2 differs from that of FHSCON.
0000007A	Error in error handling.
0000007B	The FHS save area does not have the expected structure.
0000007F	Requested actions have not been executed by DOORS.

The code xx shows how the additional return code yyyy is to be interpreted.

xx	Meaning
01 02	Parameter error; for exact cause see error code yyyy
04 05	yyyy corresponds to the FHS error MSRC with MAIN_CODE 04 for MCMAP, see page 513ff

xx	Meaning
10	yyyy corresponds to the FHS error MSRC with MAIN_CODE 16 for MCMAP, see page 513ff
20	Internal error
40	Other error; for exact cause see error code yyyy

Additional return code yyyy

yyyy	Meaning
0020	Internal error; inform systems service staff
0021	A help panel contains a field accessible to the program
0022	A command area was defined in a partial format that does not describe the lowest part of the screen.
0023	A status area was defined in a partial format that does not describe the uppermost part of the screen.
0024	A help panel was loaded for which a CCS name was agreed. This CCS name, however, does not match the CCS name of the current format.
0025 0026 0027	Internal error; inform systems service staff
0028	The format to be loaded is not of the type expected; e.g. an FHS format is to be loaded even though a KEY format was expected.
0029	Internal error; inform systems service staff
002A	Too many partial formats on a screen (more than 20)
002B	A message code is structured incorrectly
002C	An ADDPOP or REMPOP is requested without a full format being output beforehand. The previous output may have been made in line mode, or a ' non DE' format was output.
002D	Internal error; inform systems service staff
002E	Warning! A message is to be output in the message area, but the message area is too small or does not exist at all. The message is output as a box.
002F 0030 0031 0032	Internal error; inform systems service staff
0033	Partial formats are incompatible with each other; possible causes:

yyyy	Meaning
	Different format types (e.g. action format - help panel) Different column number The formats were created for different screen sizes Different color tables Different background colors Different global editing rules Editing for 8-bit terminals but for different terminal groups
0034 0036	Internal error; inform systems service staff
0037	An attempt was made to output a format as a full format, even though it was defined as a box; the POPUP control block may be missing.
0038	Internal error; inform systems service staff
0039	The format contains a nationalization which was generated by IFG as a result of the terminal specification. The MFHSCTAB module does not however contain the necessary code tables; see the section “FHS code tables” on page 497.
003A	A format not defined as a help panel in IFG, is to be output as a help panel.
003B	When combining formats, it was established that more than eight different ICE names are to be used in an image.
003C ... 0040	Internal error; inform systems service staff
0041	The data transfer area (MPUT buffer) is too short for this format.
0042	Error in IDHKHLP format; possible causes: The model format IDHKHLP for output of the KEYSHELP function cannot be found in format library. IDHKHLP was generated with CCS name; however, the displayed format does not contain a CCS name. IDHKHLP was not generated as a help panel.
0043	Internal error; inform systems service staff
0044	A KEY format cannot be found in an error handling, (error in the error handling)
0045 ... 0048	Internal error; inform systems service staff
0049	Error in the error handling (leads to FC27)
004A	Keylist not found
004F	Invalid version of map
0050	Incorrect FHS command in an implicit box (converted in message IDHS180)

yyyy	Meaning
0051	A specified field name in the ADDPOP control block or in the global attributes was not defined in the first format displayed, or the specification in the Z-CURSOR-FIELD, Z-CURSOR-INDEX or MESSAGE-LOCALIZATION is invalid.
0052	The DE format to be output in a box is too big. The maximum size of this type of format is calculated as follows: Maximum number lines = number of screen lines minus 2 Maximum number columns = number of screen columns minus 4
0053	The format is too big to be output at the required position in a box. (For the positioning, ADDPOP was specified with move thus requesting an absolute position.)
0054	Incorrect parameter in PANELID command. Neither ON nor OFF was specified (converted in message IDHS184).
0055	Incorrect parameter in KEYAREA command. Neither ON nor OFF was specified (converted in message IDHS185).
0056	Incorrect specification in SETP command (converted in message IDHS186).
0057	The paging command is not permitted (converted in message IDHS187)
0058	A help was not defined (converted in message IDHS188).
0059	The help panel was not found (converted in message IDHS191).
005A	A DE format was exchanged for a non-DE format; this is possible after restart. Response: Check format library.
005B ... 005E	Internal error; inform systems service staff
0062	Incorrect action by DOORS V2.
0064	Overflow in save area. A help can no longer be output or the number of explicit boxes is too high.
0065 0066 0067	Internal error; inform systems service staff
0068	More boxes are to be removed than were created.
0069 006A	Internal error; inform systems service staff
006B	A version not supported by FHS-DE is shown in the DE dispatcher of the format.
006C	DOORS has returned invalid specifications for a pull-down menu.
006D	A DE format required for input formatting, has been exchanged since the last output; this is possible after a restart. Response: Check format library.
006E	Internal error; inform systems service staff

yyyy	Meaning
006F	An error was found during a field check, but no message could be output in the message area or in a message box. Instead, the process is aborted (with message FC27).
0070	The exit routine requests an input field check in the global attribute; however, no exit routine was connected (see start parameters).
0071 0072 0073	Internal error; inform systems service staff
0074	Incorrect value in MODINDEX field of the list
0075	The format definition contains a comparative value that cannot be processed with the editing rule for the corresponding input field. The user is notified with message IDHS110, and the conversation is then aborted. Response: Correct format with IFG.
0076	The field attribute OUTPUT CONTROL is set to OUTPUT UNDEFINED; however, in the format the attribute “undefined“ is not permitted for this field.
0077	A mandatory input field was made protected by field attributes.
0078	The call to the system routine XHCS returned an error code. The XHCS subsystem may not be loaded.
0079	The CCS name specified in the format is not supported by the system.
007A	Internal error; inform the system service.
007B	Terminal type and device group of format not compatible.

6 FHS application in ASSEMBLER programs for DCAM/TIAM users

This chapter shows you how an FHS ASSEMBLER program must be structured and describes the FHS macros required. Not all the macros are required for #formats, and not all the operands are required with some macros; relevant notes accompany the macros concerned.

6.1 Structure of the application program

FHS macros can be called in the following types of application program for teleprocessing:

– DCAM application programs (see the “DCAM (TRANSDATA)” User Guide)
– TIAM application programs (see the “TIAM (TRANSDATA, BS2000)” User Guide)

When writing the application program you need to follow the rules for DCAM and TIAM application programs.

In the application program you have to

– generate FHS
– define the physical input/output area
– define the data transfer areas
– define a control block
– control the formatting process.

An ASSEMBLER application program using FHS can have the following structure:

```
PROGRAM  START
         MGMAP   generate mapping (i.e. call FHS)
           .
  START  BALR    assign base register
           .
         xxxxx
         MOMAP   open mapping (i.e. open FHS)
         MCMAP   call mapping ((output) formatting)
         xxxxx   send message to terminal
           .
           .
         xxxxx   fetch message into physical input/output area
         MCMAP   call mapping ((input) formatting)
         xxxxx   process data
           .           .
           .           .
         xxxxx         .
           .           .
           .           .
           .     definitions
         xxxxx   define data transfer areas (addressing aids)
           .
         xxxxx
         xxxxx   define physical input/output area
         MDCBL   define control block
           .
         MPLST   maplist
                 (only for partial formats)
         MDMEM   define memory (administrative area for 9763)
           .
         MDUSI   define user exit interface
                 (only for exit routines)
           .
           .
         END     START
```

Before each MCMAP call, the attributes and the control block can be changed and a different format application file can be specified.

6.2 Generating FHS - MGMAP macro

The macro MGMAP (generate mapping) must be called in each ASSEMBLER program that uses FHS. This macro generates FHS and reserves memory space for the directory of format definitions.

At this point you also state the access method to be used for communication and the address of the physical input/output area.

6.2.1 Description of the MGMAP macro

MGMAP - generate mapping

The MGMAP macro calls FHS and reserves memory space for the directory of format definitions.

The following must be specified in the operands of the MGMAP macro:

– the address of the physical input/output area
– the length of this area.

The following specifications are optional:

– the maximum number of format definitions present simultaneously in the address space of the application program

– the access method for communication

– the names of the format definitions that are to be loaded on opening formatting

– the name of the exit routine, and the address of the user exit interface (= exit operand block) if the latter is not specified in the associated MCMAP macro during formatting

– the name of the format application file.

Name	Operation	Operands
	MGMAP	IOAREA={adress / (r1)} ,IOLEN=n [,MAPCNT=m] [,CSTM={RTIO / DCAM}] [,RESMAP=(name1,name2,...)] [,MAPLIB=name] [,EXMOD=exitname/{exit-interface / (r2)}]

Meaning of the operands:

IOAREA= The physical input/output area defined in the application program.

In output formatting FHS sets up the device-specific message (device-specific control characters) in this area, adding a length specification to it. In input formatting FHS expects the input message and its length in this area.

address Symbolic address of the physical input/output area.

r1 Name or decimal number of a register. The register must have the address of the physical input/output area loaded before the MCMAP macro is called. Only registers 2 through 12 may be specified because MCMAP uses registers 0, 1, 14 and 15.

IOLEN=n Length of the physical input/output area.

This length must be greater than the number of character positions on the data display terminal so that the message header, any loadable character sets and the device control characters can be inserted (max. 3 control characters per data field; with 9763, up to 11 control characters when colors and character sets are used). A sufficiently large value should be specified since parts of the application program may be overwritten if the specified value is too small.

During output formatting, FHS does not check whether the physical input/output area was sufficiently large until the end. If this area was not large enough, the following area will already have been overwritten. FHS then reports an unrecoverable error. If the following area is not allocated to the program at all because the input/output area is situated at the end of the program, a dump is taken and the program is aborted. FHS cannot then even report the error.

It is therefore essential to insure that a sufficiently large physical input/output area is selected in the program. This is particularly important for the 9763 Data Display Terminal as messages may be greater than 4 Kbytes in length with this device.

MAPCNT=m Size of the directory for format definitions in main memory.

The decimal number “m” indicates the maximum number of formats that can be entered in the directory. “m” may have any value between 0 and 2730 and should be greater than the total number of all formats, subformats and character sets used.

FHS attempts to reload formats that cannot be entered in the directory because of a lack of space whenever they are used again. Actual loading takes place only on the first occurrence in this case, however.

The default value is MAPCNT=10.

Note

If MAPCNT=0 is specified, all formats are reloaded only when required. In this case, the return codes MRCF:X’0008’ and MSRC:X’0004’ do not appear in the control block.

CSTM= Access method which performs the data transfer between computer and terminal.

For output, the data format is supplied in the physical input/output area edited by FHS to match the access method specified here.

For input, FHS expects to find in the physical input/output area the data format used by the access method specified here.

The data formats for input and output used by the individual access methods are described in the appendix.

RTIO The access method is TIAM. Terminal Interactive Access Method. The system files SYSOUT and SYSDTA are assigned to a terminal. The WRTRD or WROUT macro must be specified with the operand MODE=FORM.

DCAM The access method is DCAM.

The characteristics of the connection to the terminal are stored in the Connection Control Block (CCB). When creating the CCB with the YCCB call, the operands EDIT, EDITIN and EDITOUT must be specified as follows:

```
EDIT=SYSTEM
EDITIN=(FORM,LCASE)
EDITOUT=FORM
```

HCOPY is not permitted, GETBS is optional.

The application program must prefix the length of the input message (as specified in the RPB control block) to the input message.

The length of the output message is found in the first two bytes of the physical input/output area (see MGMAP operand IOAREA) and must be transferred by the application program to the RPB control block.

RESMAP=(name1,name2,...)

The specified formats are loaded during execution of the MOMAP macro (see page 223).

The list, including parentheses, must not be longer than 127 characters.

MAPLIB=name

Name of the format application file containing the formats and subformats generated with IFG.

A format application file to be used instead of F.MAPLIB or instead of the file specified in this operand can be defined with a FILE command prior to program start.

This FILE command has the following format:

```
/SET-FILE-LINK LINK=MAPLIB,FILE-NAME=filename
```

where `filename` is the name of the format application file. If the operand is omitted, F.MAPLIB is assumed as the name of the format application file.

EXMOD=	This operand provides data relevant to an exit routine. The specified values act as preset values and can be changed in MCMAP.
exitname	Name of the exit routine (up to 8 characters)
exitinterface	Symbolic address of the user exit interface (up to 8 characters)
r2	Name (up to 8 characters) or decimal number of the register containing the address of the user exit interface.

6.2.2 Using the MGMAP macro

The following rules must be observed:

– The program must not execute the code generated by this macro.

– If formats are defined in the application program, the MGMAP macro must be called before the first format definition and in the same control section.

– If no formats are defined in the application program, i.e. if the formats were generated using IFG, the MGMAP macro may also be called in a separate module.

6.3 Controlling the formatting process

The actual process of input and output formatting is controlled in the execution section of the program.

Prior to the first formatting operation (i.e. before the first MCMAP call), the MOMAP macro must be called in order, among other things, to load the formatting routine. The MOMAP macro is described on page 223.

Input or output formatting is initiated by an MCMAP macro call. The MCMAP macro is described in the section starting on page 224.

Prior to **output formatting**, the data to be included in the format must be made available in the output data transfer area (this data transfer area must be specified in the MCMAP macro).

After the MCMAP macro has been called, the complete message is stored in the physical input/output area (see MGMAP, page 217ff).

Prior to **input formatting**, the message must be available in the physical input/output area. After the MCMAP macro has been called, the data entered in the format is located in the input data transfer area.

In addition to using the operands of the MCMAP macro, the formatting parameters can be dynamically influenced by means of the MUCBL macro (update control block) and the MATUP macro (update attributes), or by using the global and field attributes in the data transfer area with separate attribute blocks and field contents. These macros are described later in the sections starting on pages 230 and 268.

If you wish to process a format for the 9763 Data Display Terminal, the connection-specific administrative area should also be generated, using the macro MDMEM. MDMEM is described in the section starting on page 267.

The format application file can be changed during program execution by means of the MULIB macro, described in the section starting on page 228.

6.3.1 Open formatting

MOMAP - open mapping

The MOMAP macro loads the formatting routine and the formats that are specified under RESMAP in the MGMAP macro.

MOMAP stores a return code in the specified control block and in register 15. The return codes are described on page 513ff.

Name	Operation	Operands
[name]	MOMAP	{addr / (r)}

Meaning of the operands:

name Symbolic address of the macro.
“name” must not be more than 8 characters long.

addr Symbolic address of a control block defined by means of the MDCBL macro. Following execution of the MOMAP macro, this control block contains a return code which provides information on the opening sequence (see page 513ff).

r Name or decimal number of the register containing the address of the control block.

Note

MOMAP uses registers 0, 1, 14 and 15.

6.3.2 Calling formatting

MCMAP - call mapping

The MCMAP macro initiates input or output formatting.

Mandatory specifications:

– name of the required format
– address of the data transfer area
– address of the control block
– address of the restart area for #formats
– address of the MAPLIST area if a partial format is to be formatted by means of this call.
– whether input or output formatting or a service function is required

Optional specifications:

– the name of an exit routine and the address of the control block for the exit routine
– the address of the connection-specific administrative area for the 9763 Data Display Terminal

and for *formats and +formats

– the desired position of the cursor within a data field
– the address of a restart area if the program is to restore the screen contents after a screen format has been destroyed.

MCMAP stores a return code in the specified control block and in register 15. The return codes are described on page 513ff.

Name	Operation	Operands
[name]	MCMAP	{´formatname'|addr1|(r1)},{addr2|(r2)},{addr3|(r3)},{IN|OUT|SERVICE} [,EXMOD=exitname/{exit-interface|(r4)}] [,CURSOR={name|(r5)}] [,RSTARTA={restartaddr|(r6)}] [,MAPLIST={maplistaddr|(r7)}] [,MEMADR={memaddr|(r)}]

Meaning of the operands:

name Symbolic address of the macro.
“name” may be up to 8 characters long.

formatname Name of the required format (enclosed in quotes).

addr1 Symbolic address of an 8-byte field in which the format name is stored left-justified; any remainder is padded with blanks.

r1 Name or decimal number of a register containing the address of an 8-byte field. The format name must be stored left-justified in this field; the remainder of the field is padded with blanks. A subformat must not be specified here.

addr2 Symbolic address of the input or output data transfer area aligned on a halfword boundary. Halfword alignment is required only if ’aligned +formats’ are being formatted.

r2 Name or decimal number of the register containing the address of the data transfer area aligned on a halfword boundary.

addr3 Symbolic address of a control block defined by means of the MDCBL macro (see page 230ff).

r3 Name or decimal number of the register containing the address of the control block.

IN Input formatting

OUT Output formatting

SERVICE Call for an FHS service function that is specified in the macro MUCBL; applies only to #formats.

EXMOD= Specifications for the exit routine. The specifications apply to this call.

exitname Name of the exit routine (up to 8 characters).

exitinterface Symbolic address of the user exit interface (up to 8 characters).

r4 Name or decimal number of the register in which the address of the user exit interface is stored.

CURSOR= Specifies a byte of the field to which the cursor is positioned for outputs to a data display terminal; ignored in the case of #formats.

name Symbolic address of a word that holds the address of a byte in the output data transfer area.

r5 Name or decimal number of a register with the address of the byte in the output data transfer area.

Notes

– CURSOR applies only to output formatting

– If the cursor is not positioned on the first byte of a data field the position will always refer to the start address of the field in the data transfer area. The cursor will therefore not be displaced when the field is aligned.

– The cursor will be set to the field indicated in CURSOR in all instances except when “name” or the register contains 0, in which case the operand will be ignored.

– If the address indicated is invalid, the cursor will be set to the first unprotected or markable field (the first protected field on the 3270).

– If AUTOHC=YES has been specified in the MDCBL macro, then CURSOR= will be ignored (see page 232ff).

RSTARTA= Specifies the address of the restart area (aligned on a halfword boundary), which can be used for restoring destroyed screen contents. The contents of this area enables the user at any time to reconstruct the screen contents so as to display the last fully formatted screen. A specification **must** be made here for #formats.

restartaddr Symbolic address (up to 8 characters long) of the restart area

r6 Name or decimal number of a register containing the address of the restart area.

Notes

– The RSTARTA operand should not be specified for partial formats. See the section starting on page 276f for further information relating to partial formats.

– If the destroyed contents of a screen are to be reconstructed by displaying the last completely formatted screen, the address of the restart area (restartaddr) must be specified for the data transfer area (addr2).

– The RSTARTA operand is ignored if FHS finds that the formatting is to be carried out for a printer terminal. A return code to this effect is then supplied.

– In the event of a restart, all fields in the data transfer area are output in edited form.

– If MSTD=RSET was specified for output formatting, the variable fields are deleted in the restart area.

– If, during input formatting, the address of the data transfer area matches that of the restart area, FHS initiates a normal input formatting operation with EFFLEN=NO. FHS subsequently resets the EFFLEN operand to its initial value.

– The attributes “modified” and “marked” defined during input formatting are lost during a subsequent restart. There are therefore no pre-modified fields when the next input formatting operation is performed.

– If MODY=YES is set in the control block and the MCMAP operand RSTARTA= is specified, in input formatting FHS does not take the attributes from the attribute fields of the data transfer area but from the attribute fields of the restart area.

MAPLIST= Specifies the address of the work area for partial formats (MAPLIST area). This area is defined with the MPLIST macro when partial formats are used (see page 281ff).

maplistaddr Symbolic address of the MAPLIST area, as specified in the name field of the MPLST macro.

r7 Name or decimal number of the register containing the address of the MAPLIST area.

MEMADR= Provides a connection-specific administrative area when FHS is calling formatting; required only for output formatting with the 9763 Data Display Terminal.

memaddr Symbolic address of the connection-specific administrative area.

(r) Name or decimal number of the register containing the address of the connection-specific administrative area.

Note

MCMAP uses registers 0, 1, 14 and 15.

6.3.3 Updating the format application file

MULIB - update library

The MULIB macro is used during program execution to change (replace) the format application file from which the formats are then taken. The format application file remains assigned until the MULIB macro is called again.

MULIB stores a return code in the specified control block and in register 15. The return codes are described on page 513ff.

Name	Operation	Operands
[name]	MULIB	MAPLIB={´libname´ | fieldname | (r1)} ,MDCBL={cblock | (r2)}

Meaning of the operands:

name Symbolic address of the macro. “name” can be up to 8 characters long.

MAPLIB= Format application file from which the formats are to be taken

libname Name of the format application file (file name in accordance with BS2000 conventions)

fieldname Name of a 54-byte field in which the name of the format application file is stored conforming to BS2000 conventions.

r1 Name (8 characters max.) or decimal number of a register containing the address of a field, as described under “fieldname”.

MDCBL= Control block defined by the MDCBL macro.

After the MULIB macro has been executed, this control block contains the return code X’0000’ in the fields ....MRCF and ....MSRC, assuming that no errors occurred. If the control block could not be addressed, the return code X’000C’ is stored in register 15 (optional).

cblock Symbolic address of the control block

r2 Name (8 characters max.) or decimal number of a register containing the address of the control block.

Notes

– Formats that have already been called from the previously assigned format application file continue to be available.

– No formats with the same name as formats already used can be called from the new format application file; any attempt to do so would yield the format from the original format application file.

– The MULIB macro call also changes the format application file previously assigned with the command

```
/SET-FILE-LINK LINK=MAPLIB,FILE-NAME=filename
```

– MULIB uses registers 0, 1, 14 and 15.

– MULIB generates literals.

6.4 The control block

The control block is a storage area that must be defined with the MDCBL macro. It has two functions:

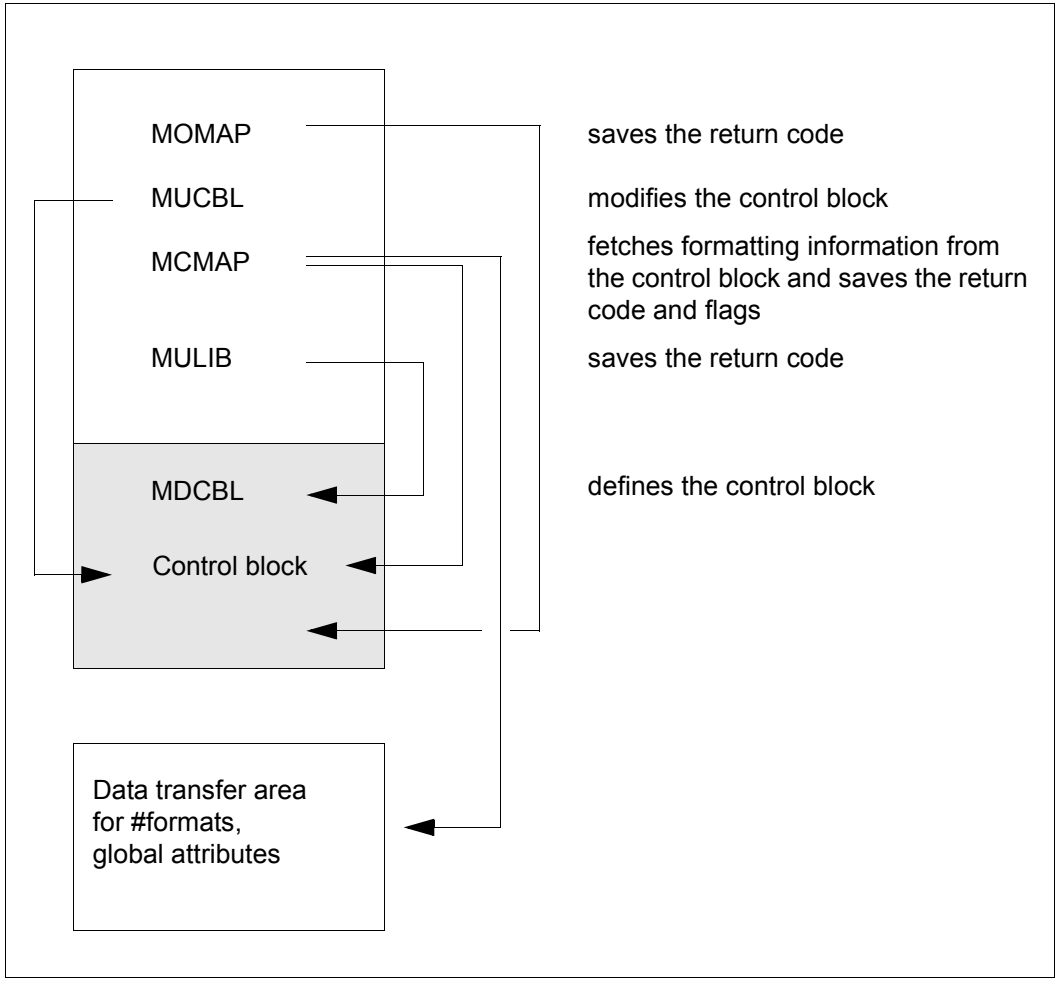
The control block is used for the definition of formatting parameters. The user specifies the appropriate operands in the MDCBL macro (see page 232ff). The entries for the formatting routine can be modified prior to each formatting run by means of a MUCBL macro call (see page 248).

The control block contains indicators (flags) as well as return codes stored by the MOMAP, MCMAP and MULIB macros. The user can address these entries in the control block. The return codes are described on page 513ff.

The control block can be defined by means of MDCBL with no operands specified. Here, formatting parameters are defined primarily by global attributes (see page 47ff). Some of the operands specified with the MDCBL macro act as start parameters for the application, i.e. they take effect only if the corresponding global attribute has the value ’DEFAULT’. Global attributes have a higher priority than start parameters, i.e. global attributes can temporarily replace start parameters (for one call).

Note

Generally one control block is sufficient, but more may be defined, e.g. to avoid the need for frequent updating. If several control blocks are specified, the first four letters of each name must be unique.



Functions of the control block

6.4.1 Defining the control block

MDCBL - define control block

The MDCBL macro is used to define the control block.

All the operands in this macro are optional. The default values are underlined in the description.

Name	Operation	Operands
name	MDCBL	[MSTD={BEGN / NLIN / PSTN / ONLY / RSET / RSON}] [,PMOD={NO / YES}] [,DETC=char1/char2] [,BEL={NO / YES}] [,MODY={NO / YES}] [,ALLATTR={NO / YES}] [,ISTD={RUNP / RMOD}] [,NILS={YES / NO}] [,EXIT={NO / YES}] [,HCOPY={NO / LOCAL / CENTRAL[/n]}] [,AUTOHC={NO / YES}] [,CLEAR={YES / NO}] [,KEYLOCK={NO / YES}] [,ATAB={YES / NO}]

Name	Operation	Operands
	MDCBL (conti- nued)	[,RESTART={NO / YES / EX}] [,MAPPART={LAST / SEGMENT}] [,HOLE={WHITE / GREY / UNDEFINED}] [,DEVICE={printer type,CNTRLU=control station / display terminal type}] [,EFFLEN={YES / NO / FLDLEN}] [,UARLEN=length] [,PAPER={0 / 1L / 1Q / 2L / 2Q / 3L / 3Q / KL / KQ}] [,HMI={1 / 2 / 3}] [,VMI={1 / 2 / 3}] [,PRNTRB={NO / YES / NEG}] [,UNLDKE={NO / YES}]

Meaning of the operands:

name Symbolic address of the control block (max. 8 characters); “name” must be specified.

MSTD= Controls how a format is output.

BEGN The complete format is output on the printer or screen.

On Data Display Terminals the complete format is output again with blanks. With printers, output is preceded by a page feed.
When using partial formatting, each partial format can be formatted with MSTD=ONLY. If the partial format has not been output before, FHS automatically formats with MSTD=BEGN.

NLIN The format is output from the beginning of the next line (’newline’). This entry applies only to printers.

PSTN Only those fields that are accessible to the program are output again on the screen, not the blanks. MSTD=PSTN presupposes that the format is already on the screen.

ONLY Output formatting:

Only those fields with contents not equal to X’00’ in the data transfer area are displayed on the screen. Thus fields containing only X’00’ are not padded with fill characters.
In the case of partial formatting, each partial format can be formatted with MSTD=ONLY even if it has not been displayed before. In this case, FHS automatically formats with MSTD=BEGN. See also the ALLATR and PMOD operands below.

Input formatting: Nothing is entered in the input data transfer area for fields with the DET attribute unless the field was detected.

RSET The last format displayed is “reset” and then output again, i.e.

- the contents of the output data transfer area are not taken into account,
- protected fields (ATTR=PROT or ATTR=PROTRET) remain on the screen unchanged,
- unprotected fields are filled with NULL characters,
- detected fields are reset and become detectable once more,
- the cursor is set to the first unprotected or detectable field on the screen (the first unprotected field with the 3270).

RSON has the combined effect of RSET and ONLY.

PMOD= Applies to Data Display Terminals only when MSTD=ONLY and only for output formatting.

NO Fields that were modified or marked during the last input are reset to “unmodified” or “unmarked”.

YES Fields that were modified or marked at the last input remain in this state. This only applies, however, to fields for which no data is received in a subsequent output formatting operation (MSTD=ONLY).

DETC= Changes the fill characters used for filling detectable fields in input formatting.

char1 A single character used for filling non-detected detectable fields in input formatting (not when MSTD=ONLY).

char2 A single character used for filling detected detectable fields.

Note

– char1 and char2 must both be specified and separated by a slash.

– char1 and char2 must not be identical.

– If the letter B is specified for char1 or char2, the blank (X’40’) is used as the fill character. If a DETC specification error is made, the default values X’00’ and X’FF’ are used as fill characters.

BEL=

NO No audible or visual alarm is triggered when the format is output to the data display terminal.

YES When the format is output to the data display terminal, an audible and visual alarm is triggered (BEL function: short beep and the “BEL” indicator on the data display terminal) - only on Data Display Terminals with a special device option. With 3270 and printers: only a visual alarm.

MODY=

NO The fields are given the attributes defined in the format.

YES The fields are given the modified attributes (see page 268ff).

ALLATTR= Controls how attribute fields are evaluated in the data transfer area for output formatting when MSTD=ONLY or RSON is set; applies only when MODY=YES.

NO Only those attribute fields are evaluated whose associated data fields have contents not equal to NULL (X’00’).

YES Every attribute field is evaluated.

ISTD= Used during output formatting to determine the input mode (READ UNPROTECTED or READ MODIFIED) for the data display terminal.

RUNP All unprotected fields and all protected fields with “automatic input” (ATTR=PROTRET) are returned to the host computer. Markable fields are not read, which means that FHS does not output them. This attribute is suppressed during output formatting.

Note on the 3270

If FHS discovers during input formatting that a field for which input is expected is missing, it makes the following entries in the data transfer area:
Missing UNPROT fields are filled with input fill characters or NULL characters. When formatting is performed with a restart, missing PROTRET fields are supplied with the field contents of the restart area. When there is no restart area, nothing is entered in the PROTRET fields.

Update outputs (MSTD=ONLY, RSET or RSON) change the attributes of fields on the screen with regard to automatic input (PROTRET) if it is not possible to output them again using the update output, as follows:

MSTD=RSET
All fields on the screen lose the automatic input attribute. PROTRET fields become PROT fields.

MSTD=ONLY and PMOD=NO
All fields output again acquire the new attributes for automatic output requested; all fields not output again lose the automatic input attribute.

MSTD=ONLY and PMOD=YES
All fields output again acquire the new attributes for automatic input requested; all fields not output again retain the automatic input attribute.

MSTD=RSON
All fields output again acquire the new attributes for automatic input; the other fields lose the automatic input attribute.

RMOD All fields modified or detected by the user, as well as any fields with “automatic input”, are sent back to the host computer and moved to the data transfer area for input. For all other fields the contents of the input data transfer area remain unchanged.

Selection fields are filled with DETC=char2 in the data transfer area if they have been detected or marked; otherwise they are filled with DETC=char1 (exception: see MSTD=ONLY).

Note on the 3270

When formatting is performed with a restart, missing PROTRET fields are supplied with the field contents of the restart area. When there is no restart area, nothing is entered in the PROTRET fields.

Update outputs (MSTD=ONLY, RSET or RSON) change the attributes of fields on the screen with regard to automatic input (PROTRET or RSET) if it is not possible to output them again using the update output, as follows:

MSTD=RSET
All fields on the screen lose the automatic input attribute. PROTRET fields become PROT fields and FSET fields become UNPROT fields.

MSTD=ONLY and PMOD=NO
All fields output again acquire the new attributes for automatic output requested; all fields not output again lose the automatic input attribute.

MSTD=ONLY and PMOD=YES
All fields output again acquire the new attributes for automatic output requested; all fields not output again retain the automatic input attribute.

MSTD=RSON
All fields output again acquire the new attributes for automatic input; the other fields lose the automatic input attribute.

NILS= Controls the handling of NULL characters (X'00') in input fields.

YES Read with NULL characters.
FHS receives the contents of fields from the data display terminal during the input transfer, correctly positioned.

Exception

On the 3270 display terminal it has the effect of NILS=NO (governed by device characteristics).

The usual rules for field alignment apply.

NO Read without NULL characters.
The data display terminal itself strips the NULL characters during the input transfer. The remaining characters are shifted to the left (implicit left alignment).

EXIT=

NO The exit routine is not called.

YES The exit routine is called.

EXIT=YES has no effect if no exit routine was specified in either the MCMAP or the MGMAP macro, or no field has the attribute “EXITROUTINE”.

HCOPY= Controls the hardcopy output of messages that are to be displayed on Data Display Terminals.
This operand additionally specifies whether the hardcopy device is attached locally or centrally - only valid for output formatting; in input formatting it is ignored.
On the 3270 display terminal HCOPY only controls the evaluation of AUTOHC.

NO No full support for hardcopy devices. AUTOHC is not evaluated for the 3270.

Note

All data fields with the attribute “printable” can be output on hardcopy devices. Output is possible from the first data field if ATAB=NO is specified, otherwise from the first modifiable data field.

LOCAL Local hardcopy support (output on a printer that is connected locally).
The data display terminal must be generated as a station with a local hardcopy printer if the information from the terminal generation is incorporated into the control block (see MUCBL DEVAR=).
AUTOHC is not evaluated for the 3270.

CENTRAL[/n] Central hardcopy support (e.g. output on a printer connected via a printer controller).
When defining HARDCOPY=CENTRAL, the number following the slash can be used to specify the channel address for the printer connected to the cluster controller. n is a number between 0 and 31.

If only HARDCOPY=CENTRAL is specified, FHS assumes channel 0 to be the printer address.
AUTOHC is not evaluated for the 3270.

Note for 8160, 975x and 9763

If HCOPY=LOCAL or CENTRAL is specified and the MUCBL operand DEVAR is not used, the cursor can be positioned on the beginning of the screen by using the keyboard, regardless of the operand ATAB, even if this screen position is protected. It is thus

possible to obtain a complete printout of the screen contents on a hardcopy device using the LA1 key (manual hardcopy). If the MUCBL operand DEVAR is used, cursor positioning via the keyboard is always dependent on the operand ATAB.

AUTOHC= Specifies whether automatic hardcopy mode is required or not.
If HCOPY=NO has been specified, this operand is ignored.

NO No automatic hardcopy mode.

In this case the terminal user is responsible for positioning the cursor if the LA1 key is actuated. The normal rules for positioning the cursor after output apply.

YES Automatic hardcopy mode.

The entire message is output automatically on the hardcopy unit. The normal rules for cursor positioning do not apply. The cursor is set to the first screen position as soon as the output is ended.
If AUTOHC=YES is specified, the CURSOR= operand is ignored in the MCMAP call.

CLEAR=

YES The screen is deleted prior to format output. For partial formatting the MAPLIST area is additionally deleted and a new device type can be defined.
The device type is defined by means of specifications in the control block (DEVICE or DEVAR) or, in the case of formats with fast formatting, through the specifications in the format definition. With partial formatting, changing the device type is permissible only when CLEAR=YES.

NO The screen is overwritten; the part of the screen not taken up by the new format remains unchanged.

Note

The CLEAR operand takes effect only for partial formatting; with full formatting, CLEAR=YES applies implicitly.

KEYLOCK= Defines the status of the keyboard on Data Display Terminals.
This operand is ignored for output on a printer.

If the printer is connected locally to data display terminal and output on the printer is required, the keyboard of the data display terminal cannot be locked by means of this operand.

NO The keyboard is not locked.

YES The keyboard is locked.

ATAB=

YES Automatic tabulator:
The cursor skips automatically from the end of an unprotected or markable field to the start of the next unprotected or markable field.
For the 3270 ATAB=YES always applies if the format was generated with IFG for application preparation for “fast formatting” and MODY=NO; when MODY=YES this applies only to fields that are not accessible to the program and gaps between fields.

NO No automatic tabulator:
The terminal user must position the cursor.
For the 3270 the field-specific ASKIP function can be implemented by means of the ASKIP attribute for fields that are accessible to the program. For fields that are not accessible to the program, the tab is controlled only by the ATAB operand. In fast formatting, ATAB=NO has no effect either on fields that are inaccessible to the program or on gaps. With the 3270 these areas are always skipped by the cursor.

MAPPART= Specifies for partial formatting whether this is the last partial formatting call for an output or not.

LAST The MCMAP call with the MAPLIST operand is the last or only partial formatting call within a partial formatting cycle.

SEGMENT The MCMAP call with the MAPLIST operand is not the last partial formatting call within a partial formatting cycle.

For more about partial formatting see page 276ff.

HOLE= Background color

WHITE Spaces between fields are displayed with normal intensity.

GREY Spaces between fields are displayed with reduced intensity.

UNDEFINED

Spaces between fields are output as protected fields which cannot be transferred. Data display terminals with default settings display the spaces with reduced intensity.

This operand is ignored for formats with ’fast formatting’. It is meaningful only for Data Display Terminals which support the function ’reduced intensity’ (9755, 9763).

RESTART= Specifies for partial formatting (value of the MAPLIST operand specified in the MCMAP call) whether restart areas are to be created for input and output formatting, or whether a restart is to be performed.

Note

The RESTART operand is only significant for the use of partial formats.

NO Partial formatting without a restart area.

YES Partial formatting with a restart area.

EX Execution of restart:
With the next MCMAP call for output formatting, the screen is reconstructed from the restart area.

Points to be observed in connection with partial formatting and restart are described in section 6.8.2.

Important

The value specified in the RESTART operand is valid for one output and the following input cycle. The operand must not be changed within one output cycle and between an output and an input cycle.

DEVICE= This specifies the type of terminal on which the format is to be output, if a different type was specified in the format definition.
The format has to be defined so that it can be output on the terminal type specified. FHS checks whether the format can be output on this terminal type (see the tables on page 536ff). If not, formatting is terminated (return code in the control block).

Restriction

When the IFG “fast formatting” function is used for format generation, optimization is carried out for the device specified in the DEVICE operand and it is not then possible, for example, to switch between data display terminal and printer.

The following entries are possible:

Type of printer:

9001	Together with the operand CNTRLU: printer terminal type.
9001-3	The number of characters per line is taken from the format definition.
9001-893	
9002	
9003	
9004	
9011-18	
9011-19	
9012	
9013	
9022	
PCL	Specification for 9021 and 9022-200 Printers
3287	Should be connected to an 8112 Printer Terminal Controller

Type of data display terminal

8160	
9750	
9751	9750 should be specified for the 9748 and 9749 Data
9752	Display Terminals, 9755 for the 9758, and 9763 for the 9756-12.
9753	
9754	
9755	
9763	
FE	Front-end terminal for using FHS-DOORS, e.g. a graphics workstation processor.
3270	If the DEVICE operand contains a valid printer terminal specification, the CNTRLU operand must also have a value that is valid. If not, the value in the format definition remains effective.

CNTRLU= Indicates the terminals on which the printers are connected as secondary peripherals.

8112 or PRCNTR — Printer is connected centrally via a printer controller.

8160L

9750L

9751L — Printer is connected locally via the specified data display terminals.

9753L

9755L

9763L

Note

CNTRLU is mandatory if you have specified a printer terminal for DEVICE. If the operand is missing, or has been entered incorrectly, the DEVICE operand is ignored and the entry taken from the format definition.

Bypass mode

Normally the DEVICE specification should correspond to the device with which the connection has been established. However, if the output is to take place on a hardcopy printer that is connected to a data display terminal with which the connection has been established without the output appearing on the screen, the printer must be specified for DEVICE and the data display terminal for CNTRLU.
A second possibility is to have a direct connection to the printer that is attached to a data display terminal and is generated as a separate station. This is not possible with TIAM.

EFFLEN= Indicates how the length fields are filled in the input data transfer area for +formats during input formatting.

YES — The length fields contain the effective length of the field in question if the field has been modified.

NO — The contents of the length field remain unchanged during input formatting.

FLDLEN — If the field was modified, the defined length of the field is entered.

Note

– The operand list for input formatting (for exit routines, EXIT=YES, see page 286ff) is only one word long when EFFLEN=NO or EFFLEN=FLDLEN is specified.

– For detectable fields (marked or unmarked) and MSTD=ONLY, EFFLEN is ignored. The length field remains unchanged. For detectable fields where MSTD ≠ ONLY, the following procedure is adopted:

  EFFLEN=YES
  The length field is set to binary zero.
  EFFLEN=NO
  The length field remains unchanged.
  EFFLEN=FLDLEN
  The defined length of the field is entered in the length field.

– For IFG formats, FHS enters the sum of the lengths of all group fields within the group when EFFLEN=FLDLEN, and when EFFLEN=YES it enters the sum of the “relevant” characters of all the group fields within the group. All characters, with the exception of the declared fill character and the NULL character, can be regarded as “relevant” characters.

UARLEN= Specifies the maximum length of the input or output data transfer area.

length A value between 1 and 32767.
If the given value is exceeded during input formatting, formatting is terminated with a return code.
In output formatting the format is processed completely. The data is taken from the data transfer area provided the specified length is not exceeded. The remaining data is formatted as if the data transfer area contained NULL characters.

Note

If the addressing aid is used as a CSECT, no value should be specified here. If it is used as a DSECT, the value must correspond to storage space available in the program.

PAPER= Specifies the type of paper feed for output on the 9003, 9004, 9011, 9012, 9013, 9022 and PCL Printers. For all other terminals the operand is ignored. The appropriate paper must be inserted in the cartridge if the operand PAPER= is to be used.

0 The format is for output on continuous stationery.

1L The format is for single-sheet feed from cartridge 1 in portrait format.

1Q The format is for single-sheet feed from cartridge 1 in landscape format.

2L The format is for single-sheet feed from cartridge 2 in portrait format.

2Q The format is for single-sheet feed from cartridge 2 in landscape format.

3L	The format is for single-sheet feed from cartridge 3 in portrait format on the 9004 Printer.
3Q	The format is for single-sheet feed from cartridge 3 in landscape format on the 9004 Printer.
KL	The format is for the form feed attachment on the 9013 Printer in portrait format.
KQ	The format is for the form feed attachment on the 9013 Printer in landscape format.

HMI= Specifies the character spacing for output on printers. For other terminals the operand is ignored.

1	Standard character spacing: 1/10 inch ≙ 2.54 mm (standard font)
2	Close character spacing: 1/12 inch ≙ 2.12 mm (condensed font 1)
3	Minimum character spacing: 1/17 inch ≙ 1.49 mm for 9001 only 1/15 inch ≙ 1.69 mm for 90xx Printers except 9001 (condensed font 2)

Refer to the table on page 536 for the maximum number of characters per line for each printer type.

VMI= Determines the line spacing for output on printers. For other terminals it is ignored.

1	Normal line spacing: 1/6 inch ≙ 4.23 mm
2	Close line spacing: 1/8 inch ≙ 3.17 mm
3	Minimum line spacing: 1/12 inch ≙ 2.12 mm

To find out how to use the remaining printer functions in your formats, consult the table on page 467.

PRNTRB= Requires a logical print acknowledgment after formatting for outputs on a printer terminal (printer return byte).

The acknowledgment is returned to the application program if the associated data display terminal is operating in bypass mode (i.e. if the format is output on the printer directly without being displayed on the terminal screen).

NO	No acknowledgment is required.

YES FHS requires a logical print acknowledgment (either positive or negative)

NEG A negative acknowledgment is required only if an error occurred during output of the format on the printer terminal.

The acknowledgment is evaluated by FHS during the next input formatting operation (see page 262ff).

Note

– The control block includes the fields ....RB1 and ....RB2, which contain the return bytes.

– If, prior to printer output formatting, the return bytes are supplied with a printable character (value between X'40' and X'FF') and PRNTRB=YES or PRNTRB=NEG is specified, these characters are sent as an acknowledgment in the cases mentioned above.

Notes on requesting printer acknowledgments

1. **Printer output with TIAM operating in bypass mode**
(no output on screen).

   – If no acknowledgments are requested (PRNTRB=NO), the format must be output using WROUT.

   – If positive and negative acknowledgments are requested (PRNTRB=YES), the format must be output using WRTRD.

   – Negative acknowledgments only (PRNTRB=NEG) are **not** permitted for output with TIAM operating in bypass mode.

2. **Printer output with TIAM operating in hardcopy mode**
(with output on screen).

   It is recommended that negative acknowledgments only or no acknowledgments be requested (PRNTRB=NO or NEG), since each acknowledgment is an input for the WRTRD call by means of which the format is output.

3. **Printer output with DCAM**

   The DCAM application must manage all input messages and thus all acknowledgments itself. Restrictions such as those for TIAM do not apply to DCAM.

In central hardcopy mode no negative acknowledgments are returned if only one printer is connected to the printer terminal controller.

UNLDKE= Controls the ejection of single sheets from the single-sheet feed or forms from the form-feed attachment.

NO Default The sheet is not ejected after printing.

YES The sheet is ejected after printing.

Note

When a form feed attachment is used on the 9013 Printer, the previous sheet must be explicitly ejected before a new sheet is inserted.

6.4.2 Updating the control block

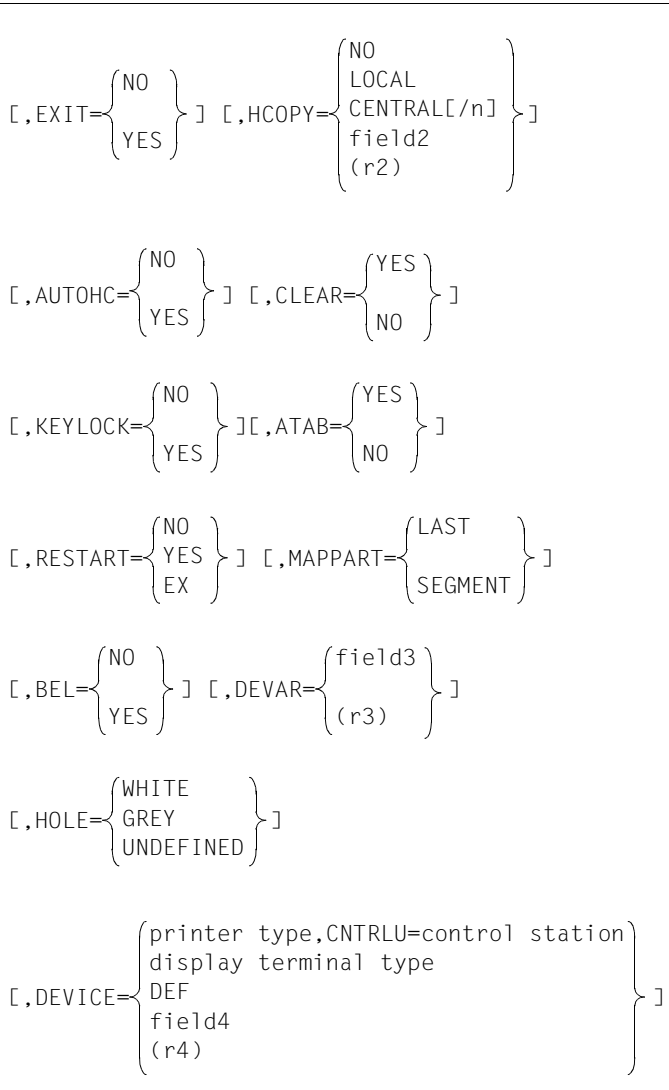
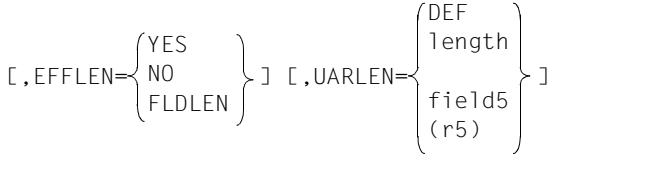
MUCBL - update control block

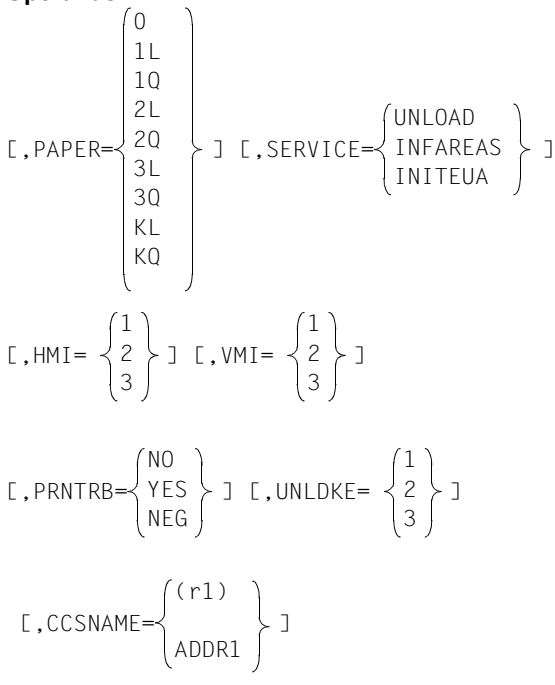
Control block values can be modified in the execution section before each MCMAP call.

The MUCBL macro alters entries in a control block already defined.
The MUCBL macro has no default values; at least one operand must be specified.

The operand DEVAR **must** be specified here for #formats, but no others if at all possible as these are specified via the global attributes.

Name	Operation	Operands
[name]	MUCBL	MDCBL={block / (register)} [,MSTD={BEGN / NLIN / PSTN / ONLY / RSET / RSON}] [,PMOD={NO / YES}] [,DETC={char1/char2 / DEF / field1 / (r1)}] [,MODY={NO / YES}] [,ALLATTR={NO / YES}] [,ISTD={RUNP / RMOD}] [,NILS={YES / NO}]

Name	Operation	Operands
	MUCBL (continue)	
	MUCBL (continue)	

Name	Operation	Operands
		

Meaning of the operands:

name Symbolic address of the macro (max. 8 characters)

MDCBL=

cblock Name of the control block to be modified.

register Name (max. 8 characters) or decimal number of a register containing the address of the control block.

DETC=

DEF — The fill characters for detectable fields are set to the default value: X'00' for undetected fields, X'FF' for detected fields.

field1 — "field1" is the name of a field containing two different characters to be used as fill characters.

(r1) — r1 is the number or name (max. 8 characters) of a register containing the name of the field in which the fill characters are stored.

HCOPY=

field2 — "field2" is a 1-byte field containing the central hardcopy address supplied by the TSTAT or YINQUIRE macro.

(r2) — r2 is the name (up to 8 bytes long) or the decimal number of a register containing the address of a field defined by HCOPY=field2.

If DEVAR has been specified, the HCOPY= operand is not evaluated.

DEVAR= Specifies an area containing information about the attached terminal. The information for this area can be obtained by means of the TSTAT macro (with TIAM) or the YINQUIRE macro (with DCAM).
This operand is mandatory for #formats. This area tells FHS

- which terminal type is connected,
- whether a hardcopy unit is connected to the terminal,
- which keyboard variant is used,
- to which control unit the terminal is connected, if the latter is a printer.

Note

- If the DEVAR operand is specified, FHS does not evaluate the DEVICE, CNTRLU and HCOPY operands.
- If the DEVAR operand is specified, and the format is intended for the locally connected printer, the following apply:

  with "fast formatting":
  the format is output in bypass mode on a local printer if it was defined for the printer (default value specified during application).

  without "fast formatting":
  the format is output to the device to which there is a connection. With TIAM this is the data display terminal and **not** the printer.

If bypass mode is required with TIAM, the data display terminal must be generated with a hardcopy printer in PDN and the hardcopy function must be set with the TCHNG command (see the manual “BS2000 User Commands (ISP Format)”).

field3 Specifies a field containing in its first 8 bytes the terminal characteristics as supplied by the TSTAT TCHAR or TSTAT ALL macro (for TIAM) or the YINQUIRE macro with OPTCD=PTNCHAR (for DCAM).

TIAM example

```
                .
                .
          MOMAP ...
                .
                .
          TSTAT TCHAR,TSTATBER,8
          MUCBL MDCBL=KONTB,DEVAR=TSTATBER
          MCMAP ...,...,KONTB,...
          WRTRD ...
          MCMAP ...,...,KONTB,...
                .
                .     P R O C E S S I N G
                .
*   D E F I N I T I O N S
KONTB     MDCBL
TSTATBER  DS    CL8
                .
                .
                .
```

DCAM example

```
                .
                .
          MOMAP
                .
                .
                .
          YINQUIRE RPB=RPB1
          MUCBL MDCBL=KONTB,DEVAR=TSTATBER
          MCMAP ...,...,KONTB,...
          YSEND ...
          YRECEIVE ...
          MCMAP ...,...,KONTB,...
                .
                .     P R O C E S S I N G
                .
```

```
*   D E F I N I T I O N S
RPB1     YRPB  AAREA=TSTATBER,OPTCD=PTNCHAR
KONTB    MDCBL ...
TSTATBER DS    CL8
           .
           .
           .
```

r3 Name or decimal number of a register with the address of a field as described in field3.

DEVICE=

FE Front-end terminal for using FHS-DOORS, e.g. a graphics workstation processor.

DEF The format is to be output on the type of terminal specified in the format definition.

field4 “field4” is a 1-byte field whose contents are supplied by the TMODE, TSTAT or YINQUIRE macro.

(r4) r4 is the name (up to 8 bytes long) or decimal number of a register with the address of a field as described in DEVICE=field4.

Notes on DEVICE

– If the TMODE, TSTAT or YINQUIRE macro specifies a terminal that is not supported, a return code is stored in the control block.

– If the terminal is specified via the TMODE, TSTAT or YINQUIRE macro, the CNTRLU operand is corrected automatically.

– Printers connected locally to a data display terminal are not supported by the TMODE, TSTAT or YINQUIRE macro. The macros only supply the type of the data display terminal in this case.

– If the DEVAR operand is specified, the DEVICE operand is not evaluated.

– A 9014 must be generated as a 9013.

UARLEN=

DEF The maximum length of the input or output data transfer area is determined by FHS.

field5 “field5” is a halfword containing, in binary, the maximum length of the input or output data transfer area.

(r5) r5 is the name (with no more than 8 bytes) or decimal number of a register containing the maximum length of the input or output data transfer area in the two low-order bytes.

SERVICE= Determines which service functions are to be called with the macro MCMAP (see page 224ff).

UNLOAD The format specified with the next 'MCMAP ...,SERVICE' call should be unloaded.

INFAREAS Initializes a function, which dynamically invokes information about the structure of the addressing aid for the current format.

FHS is provided with a description of the information to be returned by FHS as well as an indication of where the function result is to be output.

FHS continues to supply the structure of the addressing aids with each subsequent INITEUA service call until a user explicitly specifies a different structure.

You will find additional information on this service function in the section "Dynamically retrieving information on the structure of the addressing aid for #formats" on page 255.

INITEUA Initializes the data transfer area for #formats. All field attributes are supplied in accordance with their default values in the format. The global attributes remain unchanged (apart from 'formatting acknowledgments'), as do the field contents. It is thus possible to initialize data transfer areas already provided with data to the initial status. For the format whose name is specified in the first MCMAP positional operand, the data transfer area specified in the second MCMAP positional operand will be initialized if the specified format is a #format. If this is not the case, the data transfer area remains unchanged and FHS provides a return code.

r1 Decimal specification of a register with the address of an 8-byte field containing the CCS name.

All entries not explained here are described in the section dealing with the MDCBL macro.

The MUCBL macro has default values; at least one operand must be specified. Operands not modified in MUCBL retain the values specified for the MDCBL macro.

Note

Currently the MUCBL macro only uses register 14; registers 0, 1 and 15 are reserved for future developments.

The macro generates literals.

6.4.3 Dynamically retrieving information on the structure of the addressing aid for #formats

It is possible to dynamically retrieve information on the structure of the addressing aid for the current format. To do this, you invoke the service function INFAREAS to initialize processing. The area provided for the addressing aid with the next MCMAP service call must be supplied with a description of the information to be provided by FHS and an indication of where the function result is to be output.

The structure of the addressing aids is provided with each subsequent INITEUA service call until the user explicitly specifies a different structure.

Parameters for the INFAREAS service call

INFA#ELT Number of entries in the INFAELT table. This parameter should be set to X'0001', because FHS currently only returns information on the structure of the extended user area (EUA).

INFASRES This function is reserved for future enhancements and should be set to X'00'.

INFASTRL Length of table descriptor.

INFATYP Type of information required by FHS. This parameter should be set to 'ES' (EUA structure information).

INFASET Information about the output area reported to FHS:

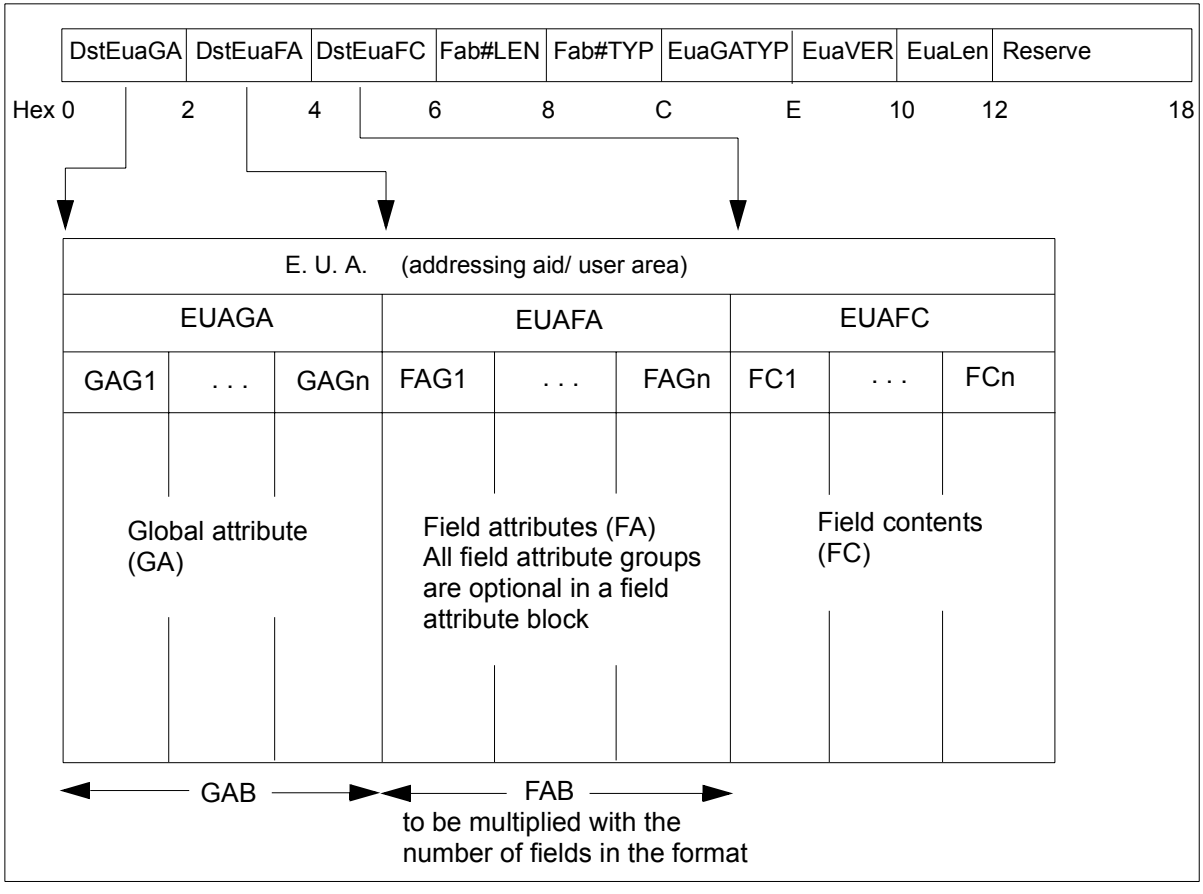
- X'01' reports that there is an area available at the address specified in INFAOBUF. At the moment, this area is filled by FHS with information about the addressing aids for the format specified in the INITEUA service call.
- X'02' reports that the area is no longer valid. FHS terminates processing.

INFAERES This parameter is reserved for future enhancements and should be set to X'00'.

INFAOBUF Address of the output area. This area must be large enough to accommodate the returned information. In the case of 'ES' (EUA structure information) it must be at least 32 bytes.

Information returned by FHS

The following addressing aid structure is supplied with each INITEUA service call:



DstEuaGA (2 Byte) Offset from the start of the addressing aids to the start of the EUA section for global attributes (EUAGA) in the addressing aids

DstEuaFA (2 Byte) Offset from the start of the addressing aids to the start of the EUA section for field attributes (EUAFA) in the addressing aids

DstEuaFC (2 Byte) Offset from the start of the addressing aids to the start of the EUA section for field contents (EUAFC) in the addressing aids

Fab#LEN (2 Byte) Length of a field attribute block (FAB)
(all field attribute areas are the same length)

Fab#TYP (4 Byte) Type of field attribute block

EuaGATYP (2 Byte) Type of global attribute block (GAB)

EuaVER (2 Byte) Version of the extended user area (to date always X'0001')
Only incremented for a structure change in a section of the previously defined extended user area

EuaLen (2 Byte) Length of addressing aids
(always provided)

Type of blocks for global attributes (GAB type)

The GAB type describes the structure of a global attribute block in the extended user area.

Bit 1 is used to mark the existence of the group *n* in the block for global attributes (EUAGA).

Bit 15 is used as an extended marking and is always 0.

Bit	Group for global attributes
1	Format confirmations Device control Output control Formatting controls
2	P keys
3	Message ID
4	Cursor variables

Defined groups for global attributes

Type of blocks for field attributes (FAB type)

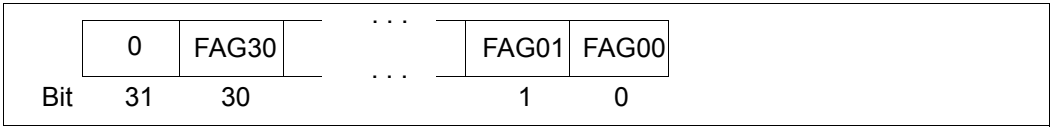
The FAB type describes the structure of a field attribute block in the extended user area.

A field attribute block (FAB) is a combination of one or more field attribute groups (FAG=Field Attribute Group).The FAB type allows 31 different field attribute groups and their combinations to be described.

It is expected that the field attribute groups provided in the FAB are sorted contiguously in ascending order.

The length of a field attribute block is the total length of all field attribute groups provided in it.

FAB type:



Each bit (n=0,...,30) of the FAB type is a marking for a specific field attribute group.

Bit n = 0 Field attribute group is **not** provided

Bit n = 1 Field attribute group is provided

Bit n = 31 Reserved as an extension marking
Provided there are not more than 31 field attribute groups, an extension is not necessary, i. e. bit 31 is always 0.

Bit	Group of field attributes
1	Field length
8	Basic attributes
9	Attribute combination
10	Field input
11	Display control
12	Field color
13	Cursor position
14	Edit return value

Defined groups for field attributes

Example

```
FHS     CSECT
FHS     AMODE ANY
FHS     RMODE ANY
        MGMAP IOAREA=IOAREA,IOLEN=4000
START   DS    0H
        USING *,10,9
        BASR  10,0
        BCTR  10,0
        BCTR  10,0
        LA    9,2048(10)
        LA    9,2048(9)
*
        TSTAT TCHAR,TSTATBER,8
        MUCBL MDCBL=BLOCK,CLEAR=INIT,DEVAR=TSTATBER
*
        MVI   INFASET,X'01'       info ON
        MUCBL MDCBL=BLOCK,SERVICE=INFAREAS
        MCMAP 'DUMMY   ',INFASTR,BLOCK,SERVICE
*
* FHS is instructed to return the information in the RETINFO area
* (specified in the INFASTR structure)
*
        MUCBL MDCBL=BLOCK,SERVICE=INITEUA
        MCMAP 'EUAFMT1 ',ADDRAIDS,BLOCK,SERVICE
*
* The addressing aids are initialized and RETINFO is filled with
* the information for EUAFMT1
*
        MCMAP 'EUAFMT2 ',ADDRAIDS,BLOCK,SERVICE
*
* The addressing aids are initialized and RETINFO is filled with
* the information for EUAFMT2
*
        MVI   INFASET,X'02'       info OFF
        MUCBL MDCBL=BLOCK,SERVICE=INFAREAS
        MCMAP 'DUMMY   ',INFASTR,BLOCK,SERVICE
*
* FHS is instructed to stop returning information
*
        MUCBL MDCBL=BLOCK,SERVICE=INITEUA
        MCMAP 'EUAFMT3 ',ADDRAIDS,BLOCK,SERVICE
*
* The addressing aids are initialized and no further
* information is returned
*
```

```
ERROR   TERM
        LTORG
BLOCK   MDCBL
*
TSTATBER DS   4H
*
INFASTR  DS    0H
INFA#ELT DC    H'1'       Number of entries in the INFAELT table
INFASRES DC    XL6'00'    Reserved for future enhancements
INFASTRL EQU   *-INFASTR  Length of table descriptor
INFAELT  DS    0H
INFATYP  DC    CL2'ES'    Type of information required *
                          ('ES' for EUA structure)
INFASET  DC    XL1'01'    Activate/Deactivate
INFAERES DC    XL1'00'    Reserved for future enhancements
INFAOBUF DC    A(RETINFO) Address of output buffer for information
INFAELTL EQU   *-INFAELT
*
RETINFO  DC    32X'00'
*
ADDRAIDS DS    0D
         DC    1000X'00'
IOAREA   DS    0D
         DC    4000X'00'
*
         END   START
```

6.4.4 Fields for return codes and flags

The control block contains the following addressable fields:

Name	Contents	Length
....MRCF	Return code	2 bytes
....MSRC	Secondary return code	2 bytes
....MRCN	General flag	1 byte
....MFZ	Function key flag	1 byte
....MKN	Short message	1 byte
....MUIL	Total length of data transfer area (in bytes)	2 bytes
....MEAL	Total length (in bytes) of data transfer area: After input formatting: Maximum length of transfer area for the format or partial format used. After output formatting: Minimum length of the transfer area for the format or partial format used for output formatting	2 bytes
....RB1	Printer return byte 1	1 byte
....RB2	Printer return byte 2	1 byte
....STB	Printer return status byte (in the device code)	1 byte
....CCSN	Name of the character set used	8 bytes
The names are formed from the first 4 characters of the control block name and the field designation. Example: The address of field ....MSRC in control block KONTB is KONTMSRC.		

The meanings of the individual return codes are given in the section starting on page 513.

6.4.5 Flags

If the terminal operator actuates one of the function or short message keys, or the input message was a printer acknowledgment, FHS stores information in the control block which can be queried in the application program.
The meaning of the codes is given in the tables which follow.

The fields ....MRCN,....MFZ,....MKN,....MUIL are used only after input formatting, ...MEAL after output formatting as well. The field ....CSSN is used after both input and output formatting.

General flags: field ....MRCN

....MRCN	Type of message
X' 00'	Data was passed to the input transfer area or a short message was received.
X' 01'	The input message was a positive printer acknowledgment.
X' 02'	The input message was a negative printer acknowledgment (for the error, see status byte ....STB).
X' 10'	No data was passed to the input transfer area.
X' 20'	The terminal operator has detected at least one field with the light pen (otherwise as for X' 00'). For the 3270 only if input was triggered by a type 1 attention field.
X' 40'	A P-key format was detected during input formatting.
X' 80'	Transfer area for output too small; the remainder of the format was output using default values.

Function key flag, field ....MFZ

....MFZ	Key on		
	9750, 9755	**9756, 9763**	**3270**
X'00'	SEND or K key		[1]
X'01'	F1		PF1
X'02'	F2		PF2
X'03'	F3		PF3
X'04'	F4		PF4
X'05'	F5		PF5
X'06'	-	F6	PF18
X'07'	-	F7	PF19
X'08'	-	F8	PF20
X'09'	-	F9	PF21
X'0A'	-	F10	PF22
X'0B'	-	F11	PF23
X'0C'	-	F12	PF24
X'0D'	-	F13	-
X'0E'	-	F14	-
X'0F'	-	F15	-
X'10'	-	F16	-
X'11'	-	F17	-
X'12'	-	F18	-
X'13'	-	F19	-
X'14'	-	F20	-
X'15'	-	F21	-
X'16'	-	F22	-
X'17'	-	F23	-
X'18'	-	F24	-

[1] ENTER key, or one of the PF keys (PF6 - PF17) mapped on Kn or one of the keys PA1 - PA3

Meaning of F1 through F24: An F key or a key mapped on F was pressed. Input data is transferred to the transfer area for input formatting.

Short message, field ....MKN

....MKN	Key in Transdata	Key in 3270
X'00'	SEND or F-Taste	[1]
X'01'	K1	PA1
X'02'	K2	PA2
X'03'	K3	PA3 or PF6
X'04'	ESC, 'V' (K4)	PF7
X'05'	ESC, 'W' (K5)	PF8
X'06'	ESC, 'M' (K6)	PF9
X'07'	ESC, 'N' (K7)	PF10
X'08'	ESC, 'O' (K8)	PF11
X'09'	ESC, '?' (K9)	PF12
X'OA'	ESC, '>' (K10)	PF13
X'0B'	ESC, '=' (K11)	PF14
X'0C'	ESC, '<' (K12)	PF15
X'0D'	ESC, ';' (K13)	PF16
X'0E'	ESC, ':' (K14)	PF17

[1] ENTER key, or one of the PF keys (PF1 - PF5, PF18 - PF24) mapped on Fn.

Meaning of K1 through K14:

A K key or a key mapped on a K key was pressed. No data is passed to the transfer area for input formatting (short message). The K2 key cannot be used in TIAM application programs; it causes a switch to system mode.

Length of data in the transfer area: field ....MUIL

Field ....MUIL is 2 bytes long; after each input formatting operation, it contains the length of the transferred data in the data transfer area. The length is dependent on the input of an end marker (EM) and on whether the format contains selection fields.

Selection field / EM	Length from start of addressing aid to	
	ISTD=RUNP	ISTD=RMOD
no selection field in the format, no EM	end of the last input field	end of the last field modified by text input
no selection field in the format, with EM	end of the last field with text before the EM	end of the last field modified by text input or EM, even if there is no text in the field
with selection fields in the format	--	end of the last field modified by text input or EM, subsequent selection fields are ignored
only selection fields in the format	--	end of the addressing aid (MUIL=MEAL)

Total length of the transfer area: field ....MEAL after input formatting

Field ....MEAL is 2 bytes long; after each input formatting operation, it contains the total length of the input transfer area for the format used.

Total length of the transfer area: field ....MEAL after output formatting

After an output formatting operation, field ....MEAL contains the minimum length of the output transfer field determined for the format used. The following special cases may occur (not with #formats):

....MEAL=X'FFFF': It was not possible to determine the minimum length of the transfer area (e.g. for MSTD=RSET since no access to the transfer area has been made in this case).

....MEAL=X'0000': It was not necessary to determine the minimum length of the transfer area (e.g. in the case of restart).

With #formats, the field always contains the length of the data transfer area in accordance with the format definition.

Printer return bytes, fields ....RB1 and RB2

Fields ....RB1 and ....RB2 are each 1 byte long and must be filled with a printable character (value between X'40' and X'FF') if acknowledgments are desired for output formatting to a printer. The user receives the contents of these bytes, if PRINTRB=YES or PRINTRB=NEG was specified in the MDCBL macro, with the acknowledgment which is returned to the application program.

If a printer acknowledgment is received (....MRCN=X'01' or X'02'), FHS enters the associated return bytes in these fields.

Printer acknowledgment status byte, field ....STB

If the input message was a printer acknowledgment, FHS enters the printer acknowledgment status byte in the device code in this 1-byte field.

Name of the character set used, ...CSSN field

With an 8-bit format, FHS enters the name of the character set used. With a 7-bit format, FHS enters blanks.

6.5 Generating the connection-specific administrative area

MDMEM - define memory

In the formatting call (MCMAP) for formats generated for a 9763 Data Display Terminal, the address of a administrative area should be specified in the case of output formatting. The device status data should be placed in this administrative area by the application program before the first formatting call. This device status data is obtained (from VTSU Version 9.0A only) by using the TSTAT macro in TIAM or YINQUIRE in DCAM, which provides information about the screen and character sets for the terminal. If no administrative area or an empty administrative area is specified in the formatting call, FHS assumes the following equipment configuration level:

– monochrome or color screen
– screen dimension 24 x 80, 27 x 132
– loadable character sets 1 through 7
– character sets 4 through 7 may also be color character sets

Note

In the case of a formatting call for formats not generated for a 9763 Data Display Terminal but which are to be output on this data display terminal type, it is nevertheless still advisable to specify an administrative area (which might be empty).

The connection-specific administrative area is used only in the case of the 9763 Data Display Terminal. With other terminals, the specification is ignored.

MDMEM generates the connection-specific administrative area and creates the symbolic start address prefix STATU. The device status data obtained using TSTAT or YINQUIRE must be stored from this point. The generated area is preset to binary zeros.

Name	Operation	Operands
[name]	MDMEM	[prefix]

Meaning of the operands:

name Symbolic name of the generated administrative area (up to 8 characters). If “name” is not specified, the default name “FHMEMORY” is generated.

prefix Generates a 1 to 3-character prefix. If “prefix” is not specified, the character string MPM is generated.

6.6 Updating attributes

This section describes the updating of attributes for formats that do **not** use the data transfer area with separate attribute blocks and field contents. For #formats, this section is of significance only when using the field attribute group 'Attribute Combination' (see page 62ff).

The MATUP macro must be called once for each data field whose attributes are to be modified during the program run. In this way the requisite data is supplied to the data field's associated attribute field (for output formatting) or length field (for input formatting) in the transfer area (see page 469).

MODY=YES must be set in the control block (MDCBL or MUCBL macro). This causes FHS to take the attributes from the attribute fields supplied with data by MATUP instead of from the format definition.

Note

In input formatting FHS expects the current attributes in the length fields; FHS returns the field length in these length fields as a function of the specifications in the MDCBL operand EFFLEN=.

Therefore

– the attribute fields and length fields must have identical contents.

– those attribute fields and length fields related to data fields whose attributes are not to be modified must be set to X'0000'.

MATUP - update attributes

MATUP supplies updated information to the attribute or length field of a data field, or to the field attribute 'Attribute Combination' in the case of #formats.

Name	Operation	Operands
	MATUP	{fieldname / (register)}, ([{UNPROT / PROT / PROTRET / FSET}][,{BRT / NORM / DRK / INVERS}] [,{PRINT / NOPRINT}][,{SIGN[ON] / DET}] [,NUM][,IC][,ITAL][,WIDE][,TALL][,ASKIP])

Meaning of the operands:

name Symbolic address of the macro

fieldname Name of the attribute field or length field to be supplied; the field attribute 'Attribute combination' is used for #formats.

register Name (max. 8 bytes) or decimal number of the register containing the address of the attribute or length field

A list of attributes can be specified to describe the characteristics of a field. If only a single attribute is specified, parentheses are not required. The CURSOR operand in the MCMAP macro can be used to move the cursor to any desired field during any formatting operation (see page 224ff).

CAUTION!

FHS inserts default values for attributes that are not specified, and not the values from the format definition (see table). At least one attribute must be specified.

Attribute		**FHS default value**
not specified	**specified**	
PRINT, NOPRINT		PRINT
PROT, UNPROT, PROTRET, FSET		UNPROT
BRT, NORM, DRK	PROT, PROTRET	NORM
	UNPROT, FSET	BRT
BRT, NORM, DRK, PROT, PROTRET, UNPROT, FSET		BRT

Protected fields - unprotected fields

UNPROT The data field is not protected. It can be overwritten on the screen and is always returned to the host computer on input when ISTD=RUNP; if ISTD=RMOD, it is returned only when the field is modified.

PROT The data field is protected. It cannot be overwritten on the data display terminal and is not returned to the host computer.

PROTRET The data field is protected and is returned to the host computer on input.

FSET In the READ MODIFIED input mode (ISTD=RMOD in the MDCBL macro; see page 232ff), the field is still returned even if it has not been modified. In the READ UNPROTECTED mode (ISTD=RUNP), FSET has the same effect as UNPROT.

Field brightness

BRT The data field is displayed at maximum brightness (default value for unprotected fields).

NORM The data field is displayed at normal brightness (default value for protected fields).

DRK The data field is invisible on the data display terminal (e.g. for entering passwords).

DRK and/or NOPRINT always produce invisible, non-printable and non-detectable fields on the 3270.

INVERS The data field is displayed in inverse video on the screen.

Hardcopy facility

PRINT The data field is printable.

NOPRINT The data field is not printable.

Flashing or light pen

SIGN The data field flashes on the data display terminal screen.

DET The data field is detectable by means of light pen or marking key. If detected, the field will flash on the screen. In input formatting the relevant field is padded in the transfer area with X'FF' characters if detected or, if undetected, with X'00' or the characters specified in MDCBL/MUCBL DETC=. DET must not be used together with PROTRET or FSET. DET is effective only in the READ MODIFIED input mode (ISTD=RMOD, MDCBL and MUCBL macros). DET is ignored in the READ UNPROTECTED input mode.

Numeric field

NUM Only numeric data can be entered in the data field, i.e. the numerics 0 (zero) through 9 (nine) and the characters * (asterisk) + (plus) , (comma) - (minus) . (period) and / (slash); on the 3270 only the numerics 0 (zero) through 9 (nine) and the characters . (period) - (minus) and the DUP key. In output formatting, however, all characters are permitted. The NUM attribute must not be specified in conjunction with PROT, PROTRET and DET, i.e. it is not permitted for protected and detectable fields.

Cursor positioning

IC The cursor is positioned in this field following data output. IC can be specified for a number of data fields. The cursor will then be set to the last field with the IC entry. IC must not be specified together with PROT or PROTRET (exception: IC with PROT and DET).

Italics

ITAL The contents of the field are represented in italics or are underlined.

Wide type

WIDE The data field will be printed in wide type on printers.

Tall type

TALL The data field is output as tall characters.

With inputs, the attributes that define the field characteristics must be the same as in the case of output.

A table showing how the display characteristics actually appear on the different terminal types is included in the appendix. If attributes specified in a format definition are not supported by the associated device, FHS substitutes default values.

How you proceed for +formats depends on whether you use a common data transfer area or separate areas for input and output formatting.

(A) Common data transfer area for input and output formatting

– Call the MATUP macro once for each field whose attributes are to be changed.

– Clear the attribute field (identical to the length field) of all data fields whose attributes are not to be changed (set them to X'0000').

On each input formatting operation, the attribute fields will be overwritten with the field length if the MDCBL operand EFFLEN has either the value YES (default) or FLDLEN.

(B) Separate data transfer areas for input and output formatting

– Call the MATUP macro twice (for the attribute field and length field) for all fields whose attributes are to be changed.

– Clear all other attribute and length fields to X'00'.

On each input formatting operation, the length fields are overwritten with the field length; the attribute fields retain their contents.

The following table shows which fields are returned to the host computer, as well as the effect of the input mode.

Attribute	**Input mode ISTD=**	
	RUNP	**RMOD**
UNPROT (not protected)	returned	returned only if modified
PROT (protected)	not returned	not returned
PROTRET (protected)	returned	returned
FSET (not protected)	returned	returned
DET with UNPROT	returned; DET is ignored	detectable; the field is given the attribute PROT
DET with PROT	not returned; DET is ignored	detectable

Notes on formats for the 3270 display terminal

– The NUM attribute is only evaluated in fields that are accessible to the program.

– The attribute combination of DET with NOPRINT and/or DRK is not possible.

– Fields with the NOPRINT and/or DRK attributes are always invisible, non-detectable and non-printable.

– Fields with the DET attribute and fields with the BRT attribute without DET are only detectable if the first character is a designator character.

  Designator characters for

  – selection fields:
    '?' becomes '>' when selected and is reset to '?' when selected again. The selection does not immediately trigger input.

– attention fields:
  Type 1 designator characters are NULL or blank. Selection triggers immediate input (field addresses of all modified fields only). All fields with the attribute BRT and NULL or blank as the first character are type 1 attention fields. However, FHS does not permit selection of such a field as data loss may occur; FHS provides a return code. The type 2 designator character is '&'. Selection triggers immediate input (field addresses and field contents of all modified fields).

You can enter designator characters for DET fields in the data transfer area prior to output formatting yourself. Note, however, that FHS treats the designator characters like regular field contents as regards justification and fill characters, i.e.

– in right justification the designator character is shifted to the right and loses its function

– in left justification the designator character disappears if it is the same as the fill character.

DET fields therefore undergo post-processing by FHS for output. After the function 'Just & Fill' and return from the exit routine for output, FHS checks the first character of the field contents. In the case of ISTD=RMOD and with the field attribute DET, only '?' or '&' is permitted as the first character; other first characters are output as '?' by FHS.

– Setting or removal of the attribute DET takes effect only if the contents of the field are output at the same time.

– For the 3270 there is an additional attribute, ASKIP.

  A field with the ASKIP attribute is automatically skipped by the cursor. When combined with DET, NUM, UNPROT, FSET or IC, ASKIP is ignored. ASKIP is only evaluated in conjunction with PROT or PROTRET. A field-specific ASKIP is only possible with ATAB=NO.

6.7 Generating attribute values

MAVAL - attribute values

The macro MAVAL generates symbolic names for the attribute values of the global attributes and field attributes of the #formats.

Name	Operation	Operands
[name]	MAVAL	[PREGA=ggg][,PREFA=fff]

Meaning of the operands:

name Symbolic address of the macro; “name” may be up to 8 characters in length.

PREGA=ggg Generates a 1 to 3-character prefix for the symbol names of the global attribute values; the preset value is PREGA=GA#.

PREFA=fff Generates a 1 to 3-character prefix for the symbol names of the field attributes; the preset value is PREFA=FA#.

6.8 The use of partial formats

For the application of partial formats FHS requires an administrative area for each terminal, the MAPLIST area where it stores information about the partial formats displayed on the screen during output and input formatting.

This MAPLIST area is defined by means of the MPLST macro. The MPLST macro and the generation of the MAPLIST area are described in the section starting on page 281.
The name of this MAPLIST area is to be specified in the MCMAP operand MAPLIST= for each partial formatting operation.

Partial formatting during output

More than one partial format can be transmitted to the terminal simultaneously. All partial formats output at the same time are formatted in one “partial formatting cycle”. Thus a partial formatting cycle consists of several MCMAP calls for partial formats to be output together. Partial formatting for #formats is described on page 78. It is controlled by way of the global attributes of the output cycle. In the case of partial formats that are not #formats, the MDCBL/MUCBL operand **MAPPART** must be specified with one of the two values **SEGMENT** or **LAST**:

SEGMENT	The following formatting call applies to a partial format within a partial formatting cycle; however, this is not the last MCMAP call in this cycle.
LAST	The following MCMAP call is the last within this partial formatting cycle.

Example

Partial formats TF1, TF2, TF3 and TF4 are to be transferred to a terminal in a single partial formatting cycle; the associated MAPLIST area is given the name MMAPLIST.

```
              .
              .
              .
            ┌MVI   GACYCCTL,GA#DEF                                    1.
optionally  └MUCBL MDCBL=KONTB,CLEAR=YES,MAPPART=SEGMENT               2.
              MCMAP 'TF1',TF1O,KONTB,OUT,MAPLIST=MMAPLIST
              BAL   7,RCODE
              MUCBL MDCBL=KONTB,CLEAR=NO
              MCMAP 'TF2',TF2O,KONTB,OUT,MAPLIST=MMAPLIST
              BAL   7,RCODE
              MCMAP 'TF3',TF3O,KONTB,OUT,MAPLIST=MMAPLIST
              BAL   7,RCODE
            ┌MVI   GACYCCTL,GA#CLOSE                                  1.
optionally  └MUCBL MDCBL=KONTB,MAPPART=LAST                            2.
              MCMAP 'TF4',TF4O,KONTB,OUT,MAPLIST=MMAPLIST
              BAL   7,RCODE
*****************************************************************
*    OUTPUT ACCORDING TO ACCESS METHOD                          *
*****************************************************************
              .
              .
              .
```

1. This statement must be issued by the user using #formats. Information is thus provided for the global attribute ’Cycle Control’.
2. This statement must be issued by the user using *formats or +formats.

Partial formatting during input

You have two options for the input formatting of partial formats:

- After the input you can format the desired partial formats displayed on the screen explicitly by means of one MCMAP call for each partial format. You specify name of the particular partial format in the MCMAP call. You can tell which partial formats are currently displayed on the screen from the MAPLIST area.
- You can also format the first partial format in which data was entered. There is no need to specify a format name in the MCMAP call for partial formatting, but the address, in a register, of a field containing only 8 blanks. FHS then supplies the name of the formatted format in this field. It is advisable to specify a “neutral” area as a data transfer area, from which data can then be transferred to the correct data transfer area, when the format name is known. Any further partial formats for which messages exist are formatted with additional MCMAP calls. For each call, the register must contain the address of a field which contains 8 blanks. For the first MCMAP call, a message is issued showing which partial format still contains modified data.
  If the message was read in with READ UNPROTECTED (ISTD=RUNP), each format can be formatted successively. If READ MODIFIED (ISTD=RMOD) is used for input, only those formats for which at least one input has been made may be formatted.

Examples

1. After an input operation, partial formats TF2 and TF4 are to be formatted (input formatting):

```
            .
            .
            .
*****************************************************************
*    INPUT ACCORDING TO ACCESS METHOD                           *
*****************************************************************
         MCMAP 'TF2',TF2I,KONTB,IN,MAPLIST=MMAPLIST
         BAL   7,RCODE
         MCMAP 'TF4',TF4I,KONTB,IN,MAPLIST=MMAPLIST
         BAL   7,RCODE
           .
           .
           .
```

2. After an input operation the first partial format containing data is to be formatted.

```
            .
            .
            .
*****************************************************************
*       INPUT ACCORDING TO ACCESS METHOD                        *
*****************************************************************
          LA    5,TFNAME
          MCMAP (5),UAFIELD,KONTB,IN,MAPLIST=MMAPLIST
          BAL   7,RCODE
            .
            .
            .
*   D E F I N I T I O N S   *
*
TFNAME  DC    XL8'40'
UAFIELD DC    1000X'00'
            .
            .
```

   After the formatting operation the field TFNAME specifies left-justified the name of the formatted partial format.

Notes on the use of partial formats

- When partial formats are output on a printer, the start line number is ignored.
- In the case of partial formatting, every partial format can be output with the MDCBL operand MSTD=ONLY even if that partial format has not been output before. FHS then changes the value internally to MSTD=BEGN to save the programmer having to access the MAPLIST area.
- If the MAPLIST operand is omitted in the MCMAP call and the format was generated as a partial format (start line number determined), a normal formatting operation is carried out which takes the start line number into account. FHS then supplies the following return codes: ....MRCF=X'0008', ... MSRC=X'2468'.
- If the format is not a partial format and the MAPLIST operand was specified in the MCMAP call, the format is output again as if with MSTD=BEGN and a warning is placed in the control block; return code ....MRCF=X'0008', ...MSRC=X'2484'.
- See also page 78 with regard to the use of partial formats that use the data transfer area with separate attribute blocks and field contents.
- Note that not all entries in the control block may be updated during one partial formatting cycle. Refer to the table below for applicable restrictions.

MDCBL operand	Modify permitted ?	Remarks
MSTD	yes	Any partial format can be output with MSTD=ONLY (instead of MSTD=BEGN). MSTD=RSET is only possible if an output formatting operation has already been carried out for the partial format.
PMOD	no	Functions as specified in first partial formatting call. Any change in subsequent partial formatting calls is ignored; a return code is output (warning).
DETC	yes	-
BEL	yes	An audible alarm is triggered for each partial formatting call with BEL=YES
MODY	yes	FHS records MODY=YES or MODY=NO in the MAPLIST area
ALLATTR	yes	-
ISTD	no	Rejected with a return code
NILS	no	Rejected with a return code
EXIT	yes	-
HCOPY	no	Functions as specified in the first partial formatting call; ignored in subsequent partial formatting calls
AUTOHC	no	Functions as specified in the first partial formatting call; ignored in subsequent partial formatting calls
CLEAR	yes	CLEAR=YES must be specified for the first partial formatting call and CLEAR=NO for subsequent calls.
KEYLOCK	no	Functions as specified in the first partial formatting call; ignored in subsequent partial formatting calls
ATAB	no	Functions as specified in the first partial formatting call; ignored in subsequent partial formatting calls
RESTART	no	Rejected with a return code
MAPPART	yes	MAPPART=LAST must be specified for the final partial formatting call and MAPPART=SEGMENT for all other calls
DEVICE	no	Rejected with a return code
CNTRLU	no	Rejected with a return code
EFFLEN	yes	-
UARLEN	yes	-
LOAD	-	Decentralized use of formats not possible for partial formatting
PAPER HMI VMI	-	Printer output without partial formatting
PRNTRB	no	Functions as specified in the first partial formatting call; ignored in subsequent partial formatting calls

6.8.1 Define MAPLIST area for partial formats - MPLST

MPLST - maplist

The MPLST macro is used to define

- in format 1, the MAPLIST area which is an administrative area required by FHS for the handling of partial formats,
- in format 2, a DSECT for that part of the MAPLIST area accessible to the program.

Format 1

Name	Operation	Operands
[name]	MPLST	

name The symbolic name of the MAPLIST area as specified e.g. in the MAPLIST= operand of the MCMAP call. If no name is specified, the area is given the name MMAPLIST.

Format 2

Name	Operation	Operands
[name]	MPLST	U [,xxx]

name Any name up to 8 characters long for the DSECT.

U Specifies that a DSECT is to be generated with this call for the user section (USER).

xxx Allows a string of up to 3 characters to be prefixed to the field name of DSECT. If no prefix is specified, the DSECT fields are given the prefix MPL.

Structure of a MAPLIST area

The MAPLIST area has the following structure:

Rel. address (hexadecimal)	Length (in bytes)	Meaning
0000	2	Length of the MAPLIST area + length of the restart area. The total length must be specified here by the application program if the restart facility is being used.
0002	2	Length of the overall area occupied after formatting (save length for UTM). This length is entered by FHS.
0004	38	MAPLIST area header
002A	2	Number of entries in the user area
002C	variable length, depending on no. of entries; 43 max.	Start address of user area; user area of the MAPLIST area containing the partial format entries. The application program may access this area by means of the DSECT generated with format 2 of the MPLST macro. This user area includes for each partial format (up to 43 are possible) a 20-byte area containing information about the partial format. The generation of this area for each partial format is described under DSECT.
---	variable length, depending on no. of entries	Internal MAPLIST area, work area for FHS

This area is defined with format 1 of the MPLST call. The area generated by the MPLST macro is always intended to accommodate 43 entries.

Structure of the DSECT for the user section of the MAPLIST area

Field name	Rel. address (hexadecimal)	Length (in bytes)	Meaning
...USLEN	00	2	Length of the MAPLIST area + length of the restart area, must be entered by the user
...FHSLN	02	2	Length of the overall area occupied after formatting
...TFUAA	04	38	MAPLIST area header
...USANZ	2A	2	Number of entries in the user area
...TFUAR	2C		Start address of the user area

This first part of the DSECT corresponds to the head of the MAPLIST area.

Field name	Rel. address (hexadecimal)	Length (in bytes)	Meaning
...FNAME	00	8	Name of partial format
...RSADR	08	4	Start address of the restart area for the partial format, relative to the start of the overall area
...RSLEN	0C	2	Length of the restart area for the partial format
	0E	2	Reserved
...UARMO	10	1	Indication of data transfer area alignment and specification in the MODY operand. This field can be interrogated with the symbolic names ...HWALN, ..., ...NOALY.
	11	3	Reserved

This second part of the DSECT corresponds to the entry made for a partial format in the user section of the MAPLIST area. Since as many as 43 partial formats can be used at a time (when each partial format consists of one line only), a maximum of 43 such entries can be made in the MAPLIST area.

This DSECT is defined with format 2 of the MPLST call.

In addition, the MPLST macro in format 2 defines the following symbolic names:

...TFUAA	MAPLIST head
...USANZ	Number of entries in the user area
...TFUAR	Start address of the user area
...LENT	Length of a partial format entry
...LUPRT	Length of the user section with 24 entries
...LUP43	Length of the user section with 43 entries
...KOPFL	Length of header section of the internal MAPLIST area
...LLIST	Length of the entire MAPLIST area

The prefix is always identified by ... in the list of symbolic names.

Notes

– The user section of the MAPLIST area should only be used as a source of information. FHS enters the information in this area. With the exception of the total length of the MAPLIST and restart areas specified in the field ...USLEN, the application program must not modify or delete field contents in the MAPLIST area.

– Programs still using the MAPLIST area in the FHS V5 structure will continue to be supported for compatibility.

6.8.2 Partial formatting and restart

If the partial formatting is to have a restart capability, the following points should be noted:

- In the MCMAP call the RESTARTA operand must be **not** be specified.
- The **RESTART=YES** operand must be specified in the MDCBL or MUCBL macro if restart areas are to be generated during partial formatting for input and output formatting. This specification is not required with #formats as the restart area is always maintained.
- The value specified in the RESTART operand (in the MDCBL/MUCBL macro) is valid for at least one output formatting cycle and the following input formatting operation. It must not be changed
  - within the current output cycle and
  - between an output cycle and a subsequent input formatting operation.
- If a restart is to be carried out, the value of the RESTART operand should be changed to RESTART=EX with a MUCBL call. In this case, the screen is reconstructed from the restart area.
  RESTART=EX only allows output formatting; the specification of OUT in the MCMAP call is mandatory.
- The restart area must immediately follow the MAPLIST area.
- Regarding the size of the restart area: :

  The restart area must be at least as large as the total length of all the addressing aids for partial formats displayed on the screen at the same time.

  \+ the length of the longest of these addressing aids, where the format configuration on the screen with the greatest total length that may occur within an application is applicable.

- Before the **first** output formatting of partial formats with a restart area, the total length of the MAPLIST and restart areas must be entered in the first two bytes of the MAPLIST area.
- After the output formatting of partial formats has been completed, FHS supplies the length of the overall area used to the application program in the field ...FHSLN of the MAPLIST area.

6.9 Checking data fields with an exit routine

Using an exit routine you can check the fields in a format for certain contents and modify them. Exit routines are not standard software, but have to be written by the user.

In the MDUSI, MCMAP, MDCBL and MGMAP macros operands must be specified.

An exit routine is separate from the application program. The exit remark will still be an EBCDIC string of characters, even if the field may contain Unicode characters. An exit routine is only executed for a field if an appropriate specification was made for this in IFG and the field is transferred. An exit routine is executed for this field during formatting after the field has been edited by the formatting routine. The exit operand block (user exit interface) provides the interface between FHS and the exit routine.

Exit routines have the following advantages:

– They can be activated whenever formatting takes place.

– The fields can be checked differently depending on their exit remark.

– The fields to be processed can be assembled into groups on the basis of their exit remark.

– The application program receives a return code set by the exit routine.

– When a field check is changed, only the exit routine needs to be modified and reassembled.

Note

When you use exit routines and a restart area, the data updates performed in the exit routine are not taken into account in the restart area in the case of output formatting.

For the effect of exit routines in the case of #formats, see page 76.

6.9.1 Operands for exit routines

What do you do to call an exit routine?

– The execution of exit routines is controlled by the application program. This is done on execution of the MCMAP macro if EXIT=YES has been entered in the control block. Exit routines can be executed during input or output formatting. Execution of the exit routine for #formats is controlled by way of the global attribute ’User Exit Control’.

– The name of the exit routine to be called must be specified in the EXMOD operand of the MCMAP or MGMAP macro.

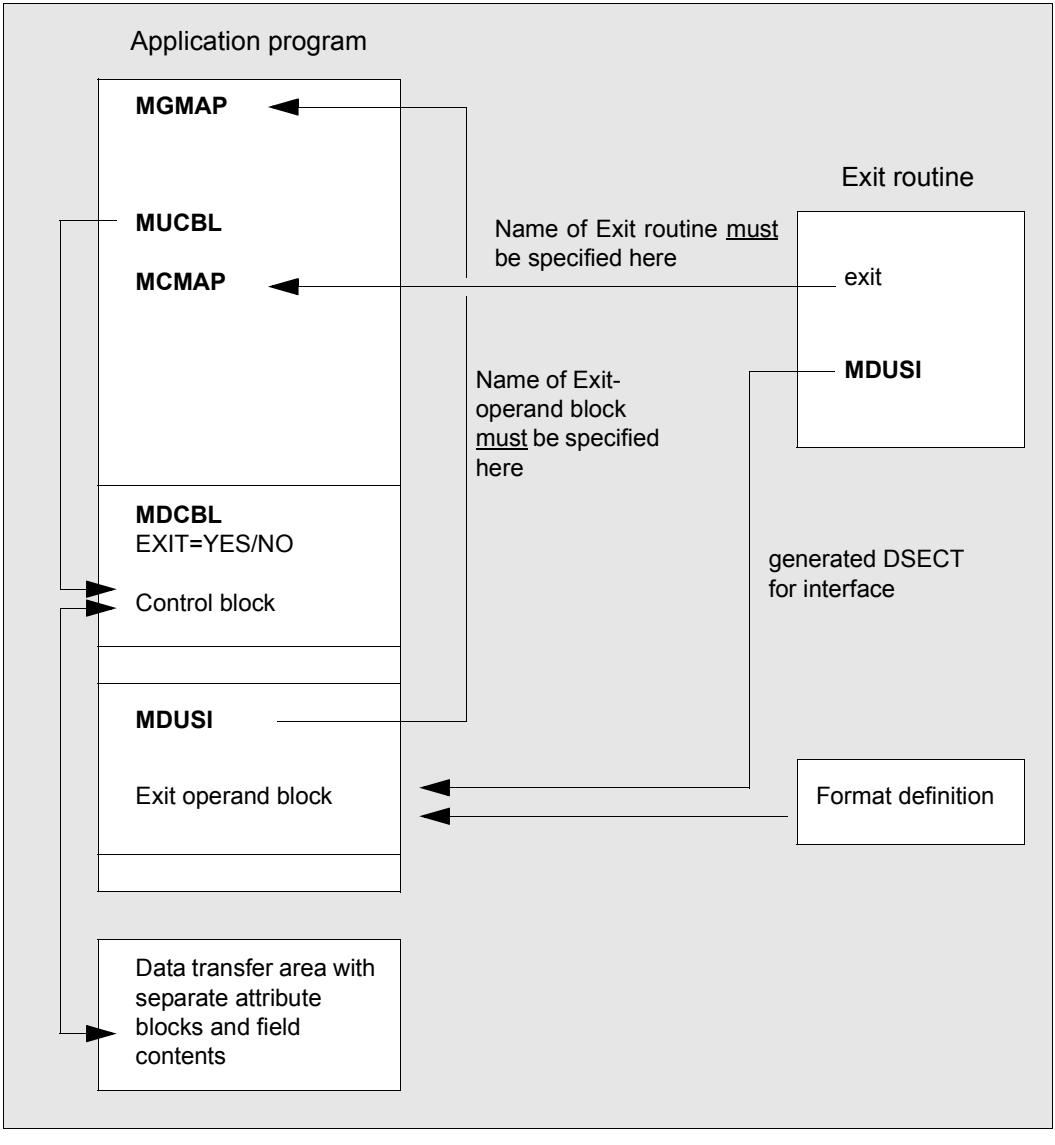
Note

The exit routine specified in the MCMAP macro is called. If no entry is present in MCMAP, the name must be specified in MGMAP.

– When you generate formats with IFG, a processing attribute is used to specify whether a field is to be checked by an exit routine. An exit remark can also be issued for the field.

– EXIT=YES (MDCBL macro) must be entered in the control block (see page 230ff) for *formats and +formats. This entry determines whether an exit routine is to be executed during a formatting run. The entry can be changed prior to each formatting operation by means of the MUCBL macro.
For #formats, it is recommended that all exit routine calls be controlled by way of the global attribute ’User Exit Control’.

– A user exit interface must be defined with the MDUSI macro (see page 292).



Overview of macros in which exit routine operands are specified

6.9.2 Creating an exit routine

In any input or output formatting operation, fields whose contents are to be checked can be processed by an exit routine. In order to transfer the data to the exit routine, the user must define an user exit interface (formerly: exit operand block). The symbolic address of this area is specified by means of the MDUSI macro (Define User Exit Interface). This macro must be inserted twice (see also page 292):

– once in the application program to define the user exit interface

– once in the exit routine as an addressing aid (DSECT) to evaluate the user exit interface defined in the application program.

User exit interface

The user exit interface is extended by the XXXXATRU flag indicating that the currently processed field is a UNICODE field. The complete information provided in the user exit interface and the structure of the block are shown below:

Field name	Field type	Field contents
XXXXREML	F	Length of the exit remark
XXXXREMC	CL8	Exit remark
XXXXINDC	CL1	Contains C' O' for output formatting or C' I' for input formatting
XXXXRECO	XL1	Return code to be passed to the application program; must be set in the exit routine
XXXXATR1	XL1	First attribute byte (see 1.)
XXXXATR2	XL1	Second attribute byte (see 2.)
XXXXATR3	XL1	Third attribute byte
XXXXATR4	XL1	Fourth attribute byte
XXXXATRU	XL1	Unicode attribute byte
	XL1	Reserved for future development
XXXXLGEN	F	Field length
XXXXLEFF	F	Effective length of the field to be handled in the input/output area
XXXXDATA	XLy	Data field; the value of the LENGTH operand of the MDUSI macro is inserted for “y”
XXXXDAT2	zX	Data field for Unicode. This field is redefined at the offset of XXXXDATA if the parameter UTF16=YES is specified in the MDUSI macro. Twice the value of the LENGTH operand of the MDUSI macro is inserted for “z”.

The first four letters of the symbolic address specified in the MDUSI macro for the user exit interface are entered for XXXX.

– If, in the XXXXATRU field, the value XXXXAU16 is set (C'U'), then the field must be handled as a UNICODE field, and the string is UTF-16

– If the field is UNICODE, then the XXXXLGEN and XXXXLEFF fields both specify a number of characters (not necessarily bytes).

– The XXXXDAT2 field will be used to access each Unicode character as a pair of bytes if needed.

– As with the previous versions of FHS, the user is responsible for specifying a buffer that is large enough to contain all characters of the largest field that may be handled by an exit routine.

1. For field XXXXATR1: :

XXXXA1UP	UNPROT attribute
XXXXA1PR	PROT attribute
XXXXA1RP	PROTRET attribute
XXXXA1NM	NUM attribute
XXXXA1DT	DET attribute
XXXXA1PT	PRINT attribute
XXXXA1IC	IC attribute
XXXXA1FS	FSET attribute

2. For field XXXXATR2:

XXXXA2SN	SIGN attribute
XXXXA2IT	ITAL attribute
XXXXA2BT	BRT attribute
XXXXA2HB	NORM attribute
XXXXA2DK	DRK attribute
XXXXA2IV	INVERS attribute
XXXXA2WD	WIDE attribute
XXXXA2TL	TALL attribute

Note

The data transferred in the XXXXATR1 and XXXXATR2 fields is purely for information purposes. Modification of this data by an exit routine has no influence on formatting.

The meaning of the attributes for the first and second attribute bytes is explained under MATUP (see page 268ff). Only current attributes can be queried.

For #formats the attribute bytes are not displayed, i.e. X'00' is supplied.

After the exit routine has been called by means of the MCMAP macro in the application program, registers 1, 13, 14 and 15 contain the following:

– Register 1: address of the operand list
– Register 13: address of the register save area
– Register 14: return address
– Register 15: address of the exit routine

The address of the user exit interface is contained in a 1- or 2-word operand list, the address of which is stored in register 1. The operand list contains one or two addresses. In the last address, the most significant bit is set to 1 as an end identifier.

In the case of +formats, the second word of the operand list contains the address of field "fieldnameL" for input formatting. This 2-byte field contains the length specification for the data field following this field. The name of the data field is "fieldnameI".

In the case of #formats and for group fields in the case of *formats, the operand list has the same structure for input and output formatting (one word in length, with the address of the user exit interface).

Example

The contents of the registers must be moved to the save area at the start of the exit routine. The save area address is stored in register 13 (see above).

```
EXROUT   CSECT
         USING *,15
         STM   14,12,12(13)
          .
          .
```

When the exit routine returns control to the application program, the register contents must be restored. To this end, the contents of the save area must be reloaded into the registers. Register 14 contains the return address for branching back to the calling module.

```
          .
          .
          .
          .
          .
          .
         LM    14,12,12(13)
         BR    14
         END
```

The exit routine is assembled separately from the application program and the resulting object module linked to the application program.

6.9.3 MDUSI macro

The MDUSI macro defines the interface for the exit routine.

Name	Operation	Operands
[name]	MDUSI	LENGTH=length[,SECT={C / D}] ,UTF16={NO / YES}

name Symbolic address of the user exit interface; “name” must not exceed 8 characters in length. The first four are entered for XXXX (see description of the user exit interface). If the “name” entry is omitted, the name USIMDUSI (with USI for XXXX) is generated.

LENGTH=length
This operand specifies the maximum field length required by an exit routine with this user exit interface. The value specified must be between 1 and 256. If this entry is omitted, an MNOTE message is output and the generation is aborted.

SECT=

C If the SECT=C operand is specified, the CSECT form of the macro is generated.

D If the SECT=D operand is specified, the DSECT form of the macro is generated.

UTF16=

YES At least one field concerned by the exit routine is a UNICODE field.

NO There is not any UNICODE field concerned by the exit routine. The macro is generated as with previous versions of FHS.

Note the following when calling the MDUSI macro:

– Value specified in the LENGTH operand must be not less than the length of the longest field in the formats for which an exit routine with this user exit interface will be called.

– The the exit routine.

– Current CSECT name will be restored after generation of the DSECT form of the macro.

Example

The format of the macro in the exit routine is:

```
EXBLOCK  MDUSI LENGTH=40,SECT=D
```

and in the application program:

```
EXBLOCK  MDUSI LENGTH=40
```

6.9.4 Example of an exit routine

The following exit routine performs different checks for fields with the exit remark A and fields with the exit remark B. Before the return branch is made, a return code is entered in the field xxxxRECO.

```
EXROUT   CSECT
         TITLE 'EXITROUTINE'
         PRINT NOGEN
         USING *,15
         STM   14,12,12(13)
         L     2,0(1)                  LAY DSECT OVER
         USING EXBLOCK,2               USER EXIT INTERFACE
         CLC   EXBLREML,=F'1'
         BNE   TESTO
         CLI   EXBLREMC,C'A'           CHECK EXIT REMARK
         BE    TESTA
         CLI   EXBLREMC,C'B'
         BE    TESTB
TESTO    MVI   EXBLRECO,C'F'           RETURN CODE FOR ERROR IN ID
         B     RETURN
TESTA    EQU   *
****************************************
*        CHECK FIELD WITH REMARK A      *
*                                       *
*        SET RETURN CODE EXBLRECO       *
****************************************
         B     RETURN
TESTB    EQU   *
****************************************
*        CHECK FIELD WITH REMARK B      *
*                                       *
*        SET RETURN CODE EXBLRECO       *
****************************************
RETURN   EQU   *
         LM    14,12,12(13)
         BR    14
*
*
EXBLOCK  MDUSI LENGTH=15,SECT=D        DSECT FOR USER EXIT INTERFACE
         END
```

6.10 Example of an ASSEMBLER application program using FHS

The example refers to the format below.

```
DELIVERY NOTE
Sender
Last name:  ____________________
First name: ____________________
Street:     ________________________
ZIP:        #####
City:       ________________________
Telephone  Code:  #####    Number:  #######

Recipient
Last name:  ____________________
First name: ____________________
Street:     ________________________
ZIP:        #####
City:       ________________________
Customer no.:  ##########
```

```
PRGASS   START
         PRINT NOGEN
*
*  DATA TRANSFER AREA                                (+FORMAT)
*       NOT ALIGNED, WITH ATTRIBUTE FIELDS
*
*  PROGRAM
*       PERFORMS ONLY OUTPUT/INPUT OF THE FORMAT
*       (NO PROCESSING OF FIELDS !)
*
*
*
*
*  READY FHS                                                  *
         MGMAP IOAREA=INOUT,                                   -
               IOLEN=2048,                                     -
               CSTM=RTIO,                                      -
               MAPLIB=$C.MANUALEXAMPLES.LMSLIB
*
*
*************************************************************************
* PROGRAM START                                                         *
*************************************************************************
*
*
```

```
ANF      BALR  3,0
         USING *,3,4
         LA    4,4095(3)
         LA    4,1(4)
R1       EQU   1
R7       EQU   7
R14      EQU   14
*
*  OPEN FHS                                                    *
         MOMAP CONTROL
*
*  CALL FORMATTING                                             *
CALL     MCMAP 'FORMASS',FORMASSO,CONTROL,OUT
         BAL   R7,RCODE
         WRTRD INOUT,,INOUT,,2048,ERROR,MODE=FORM
         BAL   R7,RCODE
         MCMAP 'FORMASS',FORMASSI,CONTROL,IN
         BAL   R7,RCODE
*
*  PROGRAM END                                                 *
STOP     TERM
*
*  TRANSFER AREA FOR OUTPUT AND FOR INPUT                      *
         FORMASSO
         FORMASSI
*
*  ERROR IM 'WRTRD'                                            *
ERROR    EQU   *
         WROUT 'ERROR IM WRTRD',STOP
         TERMD
*
*  CHECK RETURN CODE                                           *
RCODE    EQU   *
         CLC   KONTMRCF(2),=X'0000'
         BE    OK
         TERMD
OK       BR    R7
*
*  DATA DEFINITIONS                                            *
INOUT    DS    0H
         DS    2048CL1
INOUTL   EQU   *-INOUT
*
*  CONTROL BLOCK                                               *
CONTROL  MDCBL MSTD=BEGN,                                       -
               MODY=NO,                                         -
               ISTD=RUNP
         END   ANF
```

7 FHS application in COBOL programs for DCAM/TIAM users

This chapter describes the use of formats with the FHS COBOL interface.

7.1 Introduction to the FHS COBOL interface

The FHS COBOL interface enables the COBOL programmer to implement the full range of FHS functions in TIAM or DCAM application programs without having to write an ASSEMBLER subprogram for the formatting.
The formatting process has been integrated into the calls of the access method. Before the input/output call, you merely have to supply certain data structures with data. FHS obtains formatting parameters from these structures and stores return codes in them. Formatting is possible using the following DCAM or TIAM calls:

TIAM calls (see page 343ff):

– CALL “WROUT”
– CALL “WRTRD”

DCAM calls (see page 352ff):

– CALL “YSEND”
– CALL “YRECEIVE”

In addition, you can make use of the following FHS COBOL calls, implemented in the form of subprogram calls (CALL ...):

– CALL “FHSCURS” for selectively positioning the cursor prior to output (see page 361ff)

– CALL “FHSATTR” for updating field attributes (see page 363ff)

– CALL “FHSINIT” for initiating the formatting and defining certain start parameters for formatting (see page 366ff).

– CALL “FHSSERV” for calling special FHS service functions (see page 370ff).

Except in the case of the FHSATTR call, it is possible to modify attributes by means of the copy element FHSATTRM and a simple MOVE statement (see page 381ff).

You generate the formats using IFG.

The data structures used by FHS COBOL are provided in the form of copy elements. This simplifies the transfer of the formatting parameters to FHS. These data structures are described in the section starting on page 300.
Notes on the application of partial formats are given in the section starting on page 395.
Points to watch when compiling and linking FHS COBOL programs are given in the section starting on page 402.

FHS COBOL programs have the following structure:

```
   DATA DIVISION.
    WORKING-STORAGE SECTION.
        01   name1.
        COPY FHSMAINP.                          ready FHS-MAIN-PAR

       [01   name2.   ]
       [COPY FHSATTRP.]                         [ready FHS-ATTR-PAR]

       [01   name3.   ]
       [COPY FHSATTRM.]                         [COPY-members for MOVE]

       [01   name4.   ]
       [COPY FHSINITP.]                         [ready FHS-INIT-PAR]
                .
                .
-------------------------------------------------------------------------------
    PROCEDURE DIVISION.
       [CALL „FHSINIT“ USING FHS-CONTROL-INFO
                             FHS-INIT-PAR       [initiate formatting]
                             area
                             [format-list].]
                               .
                .
       [CALL „FHSCURS“ USING FHS-MAIN-PAR       [position cursor]
                             fieldname.]
                .
                .
       ***************************************************************
       *                                                             *
       *   Formatted input/output in accordance with the access method   *
       *                                                             *
       ***************************************************************
                .
                .
       [CALL „FHSATTR“ USING FHS-CONTROL-INFO
                             FHS-ATTR-PAR       [attribute update with
                             attributfeld.]      FHSATTR call or
                o r                              MOVE statement]
       [MOVE KCxxxx TO attribute-field.]
                .
                .
```

Structure of an FHS COBOL program

Optional entries appear in brackets.

7.2 Data structures used by FHS COBOL

Data structures provide the interface between FHS and the application program; the data structures are stored as copy elements in a library and copied from this library into the application program. They are specified in the COBOL program under USING in the CALL macro.

These data structures have the following functions:

– They contain the formatting parameters which are passed by the application program to FHS.

– The areas are evaluated by FHS and default values are assumed where entries are missing.

– FHS stores information in them (e.g. return codes) about the execution of the FHS COBOL call.

In some cases (e.g. in order to avoid frequent changes), it is necessary to define these data structures more than once; if this is the case, a tabular form should be used.

You copy the data structures into the program as follows:

– If the data structure is required only **once**, copy it into the program with

```
01   name.
     COPY data-structure.
```

You have to assign the level number (01) yourself.

– If the data structure is to be defined **more than once**, copy it into the program with

```
01    table.
   02 name OCCURS i.
      COPY data-structure.
```

The following data structures are available:

FHS-MAIN-PAR

This data structure is divided into two parts.

In “FHS-CONTROL-INFO”, the application program receives information about the formatting run (e.g. return codes).
The application program controls the formatting run by means of “FHS-MAP-PAR”.
The FHS-MAIN-PAR data structure is copied into the application program with the COBOL statement

```
01    name
      COPY FHSMAINP.
```

FHS-INIT-PAR

This data structure can be used by the application program in the FHSINIT call to supply its own default values for the subsequent formatting operations and to specify the format application file.
The FHS-INIT-PAR data structure is copied into the application program by means of the COBOL statement

```
01    name
      COPY FHSINITP.
```

FHS-ATTR-PAR

This data structure is required for the modification of attributes with the FHSATTR call for formats that do not use the data transfer area with separate attribute blocks and field contents.
It is copied into the application program with the COBOL statement

```
01    name
      COPY FHSATTRP.
```

FHS-EXITMOD-PAR

This data structure corresponds to the user exit interface (see page 289ff). It is only required when exit routines are used. It is copied into the LINKAGE SECTION of the exit routine by means of the COBOL statement

```
01    name
      COPY FHSEXITP.
```

This data structure is part of FHS-MAIN-PAR in the application program.

In addition, FHS also provides the data structure “FHS-ATTRIBUTE-MOVE”, which is copied into the application program with the COBOL statement

```
01    name
      COPY FHSATTRM.
```

This allows an FHSATTR call to be replaced by a simple MOVE statement whenever there is a suitable combination of attributes in this data structure.

FHS-ATTRIBUTE-VALUES

This list generates symbolically addressable attribute values for the global attributes and field attributes of the data transfer area with separate attribute blocks and field contents. It is copied into the application program by means of the COBOL statement

```
COPY FHSAVAL
```

FHS-CCSN-PAR

The “FHSSERV” call requires this data structure to receive the name of the character set used in the format. It is copied into the application program by means of the COBOL statement

```
COPY FHSCCSNP
```

The following table shows which data structures are required for which FHS COBOL calls:

Data structure	used in						
	WROUT WRTRD	**YSEND YRECEIVE**	**FHSINIT**	**FHSATTR**	**FHSCURS**	**FHSSERV**	**Exit-routine**
FHS-MAIN-PAR	X[1]	X	X[2]	X[2]	X	X	
FHS-INIT-PAR			X				
FHS-ATTR-PAR				X			
FHS-EXITMOD-PAR							X
FHS-CCSN-PAR						X	

[1] X = The data structure is used in the call.

[2] The call uses only the part of the data structure containing the return codes.

7.2.1 The FHS-MAIN-PAR data structure

FHS-MAIN-PAR is the name of the following data structure, which is copied into the application program by means of “**COPY FHSMAINP**”.

Note

In the case of #formats, global attributes often take effect instead of FHS-MAIN-PAR.

```
 01  fhs-work-area.
     COPY FHSMAINP.
*****************************************************************
* NAME              FHSMAINP                                    *
* VERSION           811                                         *
*                                                               *
*                       /->  FHS-CONTROL-INFO                   *
*       FHS-MAIN-PAR -                                          *
*                       \->  FHS-MAP-PAR                        *
*                                                               *
* END-INTERFACE     FHSMAINP                                    *
*****************************************************************
 35  FHS-MAIN-PAR.
     41  FILLER                      PIC 9(5) COMP SYNC.
     41  FILLER                      PIC X(52).
*
 35  FHS-CONTROL-INFO      REDEFINES  FHS-MAIN-PAR.
*
     41  FHS-MAIN-RC                 PIC 9(4) COMP SYNC.
     41  FILLER                      PIC X(6).
*
     41  FHS-ERROR-INFO.
         42  ERROR-CATEGORY          PIC 9(4) COMP SYNC.
         42  ERROR-REASON            PIC 9(4) COMP SYNC.
     41  FILLER                      PIC X(7).
     41  PRINTER-RETURN-MSG.
         42  RETURN-MSG-TYPE         PIC X.
         42  RETURN-BYTE1            PIC X.
         42  RETURN-BYTE2            PIC X.
         42  RETURN-STATUS-INFO      PIC XX.
      *
           41  FHS-OUTPUT-INFO.
         42  FILLER                  PIC X(11).
         42  OUT-USER-AREA-TRUNCATION PIC X.
         42  OUT-USER-AREA-LEN       PIC S9(5) COMP SYNC.
```

```
*
     41  FHS-INPUT-INFO.
         42  FILLER                    PIC X.
         42  IN-PRINTER-RETURN-MSG     PIC X.
         42  IN-FIELD-DET              PIC X.
         42  IN-MSG-NILS               PIC X.
         42  IN-F-KEY                  PIC 9(2) COMP SYNC.
         42  IN-K-KEY                  PIC 9(2) COMP SYNC.
         42  IN-USER-AREA-LEN          PIC 9(5) COMP SYNC.
         42  IN-MSG-LEN                PIC 9(5) COMP SYNC.
*
*
 35  FHS-MAP-PAR.
*
     41  FHS-MAP-PAR-GENERAL           PIC X(160).
*
     41  FHS-MAP-GENERALS  REDEFINES  FHS-MAP-PAR-GENERAL.
         42  FHS-MAP-NAME              PIC X(8).
         42  FHS-EXIT-MOD-NAME         PIC X(8).
         42  FHS-MAPPING-METHOD        PIC X(4).
         42  FHS-MODY-ATTRS            PIC X.
         42  FHS-PARTIAL-MAP-OPT       PIC X.
         42  FHS-MAP-PART              PIC X.
         42  FHS-MAP-CURSOR-OPT        PIC X.
         42  FILLER                    PIC X(4).
         42  FILLER                    PIC X(4).
         42  FHS-SERVICE-FUNCTION      PIC 9(3) COMP SYNC.
         42  FHS-RESTART-OPT1          PIC X.
         42  FHS-RESTART-OPT2          PIC X.
         42  FHS-MAP-LIB-OPT           PIC X.
         42  FHS-MAP-LIB-NAME          PIC X(54).
         42  FHS-EXIT-LIB-OPT          PIC X.
         42  FHS-EXIT-LIB-NAME         PIC X(54).
         42  FHS-EXIT-FOR-OUTPUT       PIC X.
         42  FHS-EXIT-FOR-INPUT        PIC X.
         42  FHS-DESIRED-CCSNAME       PIC X(8).
         42  FILLER                    PIC X(4).
*
     41  FHS-MAP-PAR-OPTIONAL          PIC X(60).
*
     41  FHS-MAP-OPTIONS   REDEFINES  FHS-MAP-PAR-OPTIONAL.
         42  MAP-DEVICE-CLASS          PIC X(4).
         42  MAP-PRINTER-CONTROL       PIC X(4).
         42  FILLER                    PIC X(4).
         42  MAP-AUTO-TAB              PIC X.
         42  MAP-EFF-LEN               PIC X.
         42  MAP-POS-DET-CHAR          PIC X.
```

```
           42  MAP-NEG-DET-CHAR         PIC X.
           42  FILLER                   PIC X(8).
           42  MAP-READ-METHOD          PIC X(4).
           42  MAP-SCREEN-PRE-MOD       PIC X.
           42  MAP-READ-NILS            PIC X.
           42  MAP-USE-ALL-ATTRS        PIC X.
           42  MAP-PRINTER-OPTION       PIC X.
           42  MAP-PRINTER-RETURN-BYTE1 PIC X.
           42  MAP-PRINTER-RETURN-BYTE2 PIC X.
           42  FILLER                   PIC X.
           42  MAP-HARDCOPY-OPTION      PIC X.
           42  FILLER                   PIC XX.
           42  MAP-AUTO-HARDCOPY        PIC X.
           42  MAP-LOCK-KEYS            PIC X.
           42  MAP-CLEAR-OPTION         PIC X.
           42  MAP-BEL-OPTION           PIC X.
           42  MAP-PRINT-FORMAT-OPTION.
               43  MAP-PRINT-LINES      PIC X.
               43  MAP-PRINT-COLUMNS    PIC X.
               43  MAP-PRINT-PAPER      PIC X.
               43  MAP-PRINT-FORM       PIC X.
           42  MAP-LIB-LOAD-OPTION.
               43  MAP-LIB-LOAD-MODE    PIC X.
               43  MAP-LIB-LOAD-FILE    PIC X.
           42  MAP-HOLE-COLOR           PIC X.
           42  FILLER                   PIC X(11).
*
       41  FHS-EXIT-PAR.
           42  EXIT-IDENT-LEN           PIC 9(5) COMP SYNC.
           42  EXIT-IDENT               PIC X(8).
           42  EXIT-IN-OUT              PIC X.
           42  EXIT-RET-INFO            PIC X.
           42  FILLER                   PIC XX.
           42  FILLER                   PIC X(4).
           42  EXIT-FLD-LEN             PIC 9(5) COMP SYNC.
           42  EXIT-EFF-LEN             PIC 9(5) COMP SYNC.
            42  EXIT-DATA                PIC X(80).
```

In the case of a Unicode application, FHS-MAIN-PAR must be copied in the application program by means of “COPY FHSMAINU”. It differs from FHSMAINP by the definition of the FHS-EXIT-PAR data structure:

```
41  FHS-EXIT-PAR.
    42  EXIT-IDENT-LEN          PIC 9(5) COMP SYNC.
    42  EXIT-IDENT              PIC X(8).
    42  EXIT-IN-OUT             PIC X.
    42  EXIT-RET-INFO           PIC X.
    42  FILLER                  PIC XX.
    42  FILLER                  PIC X(2).
    42  EXIT-U-FLAG             PIC X.
    42  FILLER                  PIC X.
    42  EXIT-FLD-LEN            PIC 9(5) COMP SYNC.
    42  EXIT-EFF-LEN            PIC 9(5) COMP SYNC.
    42  EXIT-DATA-U             PIC N(132).
    42  EXIT-DATA REDEFINES EXIT-DATA-U PIC X(264).
```

Description of the data items

FHS-CONTROL-INFO
contains information and return codes from FHS for the application program.

FHS-MAIN-RC
This item is where FHS writes the primary return code. If the code is 0, the FHS call was error-free. For the meaning of the remaining entries, see page 524ff (return codes).

FHS-ERROR-INFO
This is where FHS writes the secondary return code. If the FHS call was processed without errors, the code is 0. Otherwise,

ERROR-CATEGORY
contains the error type, and

ERROR-REASON
cause of the error.
For the precise meaning of the individual entries, see page 524ff (return codes).

PRINTER-RETURN-MSG
contains acknowledgment information after a logical print acknowledgment has been received from the printer, otherwise the value is LOW-VALUE. The IN-PRINTER-RETURN-MSG item in FHS-INPUT-INFO specifies whether the input message is a printer acknowledgment. The printer acknowledgment is provided in the following items:

RETURN-MSG-TYPE
specifies whether it is a positive or negative printer acknowledgment:

“+” The input message is a positive printer acknowledgment.

“-” The input message is a negative printer acknowledgment.

RETURN-BYTE1

RETURN-BYTE2
In these items FHS returns the values the items MAP-PRINTER-RETURN-BYTE1 and MAP-PRINTER-RETURN-BYTE2 at the time of output formatting.

RETURN-STATUS-INFO
contains the status byte of the acknowledgment (in hexadecimal notation). The status byte is returned in device code since each bit has a particular significance. For the meaning of these values see the reference manual for your printer.

FHS-OUTPUT-INFO
contains output information following an output formatting operation.

OUT-USER-AREA-TRUNCATION
specifies whether the data transfer area for output is large enough.

LOW-VALUE when size sufficient

HIGH-VALUE when too short; the remainder of the format is output using default values.

OUT-USER-AREA-LEN
specifies a number of bytes.
In the case of #formats, always contains the length of the data transfer area according to the format definition after an output formatting operation. With +formats and *formats, this data item contains the minimum length computed for the data transfer area after an output formatting operation, if computed by FHS.

The entries in this item have the following meaning:

OUT-USER-AREA-LEN > 0
Minimum length computed for the data transfer area.

OUT-USER-AREA-LEN = 0:
The minimum length of the data transfer area did not need to be computed (e.g. in the event of a restart).

OUT-USER-AREA-LEN < 0:
The minimum length of the data transfer area could not be computed since it was not necessary to access the data transfer area during output formatting (e.g. with FHS-MAPPING-METHOD = “RSET”).

FHS-INPUT-INFO
contains information concerning input following an input formatting operation. This information is provided in the following items:

IN-PRINTER-RETURN-MSG
specifies whether the input message is a printer acknowledgment.
The contents of this acknowledgment are supplied in the item PRINTER-RETURN-MSG.

LOW-VALUE The input message is not a printer acknowledgment.

HIGH-VALUE The input message is a printer acknowledgment.

IN-FIELD-DET

indicates whether a field was detected:

LOW-VALUE No field was detected.

HIGH-VALUE One or more fields were detected.

IN-MSG-NILS indicates whether data was received:

LOW-VALUE The data transfer area for input formatting contains the received data or a short message was received.

HIGH-VALUE The data transfer area for input formatting does not contain any data, nor was any short message received, or a P-key format was detected during input formatting (see page 43).

IN-F-KEY

This item contains the function key display.

....MFZ	Key on		
	9750, 9755	**9756, 9763**	**3270**
X'00'	DUE or K key		[1]
X'01'	F1		PF1
X'02'	F2		PF2
X'03'	F3		PF3
X'04'	F4		PF4
X'05'	F5		PF5
X'06'	-	F6	PF18
X'07'	-	F7	PF19
X'08'	-	F8	PF20
X'09'	-	F9	PF21
X'0A'	-	F10	PF22
X'0B'	-	F11	PF23
X'0C'	-	F12	PF24
X'0D'	-	F13	-
X'0E'	-	F14	-
X'0F'	-	F15	-
X'10'	-	F16	-
X'11'	-	F17	-
X'12'	-	F18	-

....MFZ	Key on		
	9750, 9755	**9756, 9763**	**3270**
X'13'	-	F19	-
X'14'	-	F20	-
X'15'	-	F21	-
X'16'	-	F22	-
X'17'	-	F23	-
X'18'	-	F24	-

[1] ENTER key or one of the keys mapped onto Kn (PF6 - PF17) or one of the keys PA1 - PA3 or attention field

Meaning of F1 through F24:
An F key or key mapped on F was pressed. Input data is passed to the data transfer area for input formatting.

IN-K-KEY When a short message is received, this item contains the corresponding code.

....MKN	Key in TRANSDATA	Key on 3270
X'00'	SEND or F Key	[1]
X'01'	K1	PA1
X'02'	K2	PA2
X'03'	K3	PA3 or PF6
X'04'	ESC, 'V' (K4)	PF7
X'05'	ESC, 'W' (K5)	PF8
X'06'	ESC, 'M' (K6)	PF9
X'07'	ESC, 'N' (K7)	PF10
X'08'	ESC, 'O' (K8)	PF11
X'09'	ESC, '?' (K9)	PF12
X'OA'	ESC, '>' (K10)	PF13
X'0B'	ESC, '=' (K11)	PF14
X'0C'	ESC, '<' (K12)	PF15
X'0D'	ESC, ';' (K13)	PF16
X'0E'	ESC, ':' (K14)	PF17

[1] ENTER key, or one of the PF keys (PF1 - PF5, PF18 - PF24) mapped on Fn, or an attention field.

IN-K-KEY ≠ 0 means:

A K key or key mapped on K was pressed. No data is passed to the data transfer area for input formatting (short code). In TIAM application programs the K2 cannot be used; there it causes a switch to system mode.

IN-USER-AREA-LEN

specifies a number of bytes.
This item contains the overall length of the data transfer area for the entire format after an input formatting operation.

IN-MSG-LEN

specifies a number of bytes.
This item contains the length of the transferred data in the data transfer area after each input formatting operation. The length is dependent on the input of an end marker (EM) and on whether the format contains selection fields.

Selection field / EM	**Length from start of addressing aid to**	
	MAP-READ-METHOD="RUNP"	**MAP-READ-METHOD="RMOD"**
no selection field in the format, no EM	end of the last input field	end of the last field modified by text input
no selection field in the format, with EM	end of the last field with text before the EM	end of the last field modified by text input or EM, even if there is no text in the field
with selection fields in the format	--	end of the last field modified by text input or EM, subsequent selection fields are ignored
only selection fields in the format	--	end of the addressing aid

FHS-MAP-PAR

This part of the FHS-MAIN-PAR data structure is used by the application program to control the formatting. In such cases, the format name must be entered in FHS-MAP-NAME.

FHS-MAP-PAR-GENERAL

The part of FHS-MAP-PAR to which the application program transfers general information concerning the formatting (e.g. format name).

FHS-MAP-NAME

This item is where the application program enters the name of the format definition to be used for input/output. This item **must** be filled.

FHS-EXIT-MOD-NAME

This item is where the application program enters the name of the module containing the exit routine being used.

FHS-MAPPING-METHOD

This item determines the type of formatting for output. It is ignored for #formats.

The following entries are possible:

"BEGN" The complete format is output.

On Data Display Terminals, the complete format is redisplayed with blanks. On printer terminals, output is preceded by a feed to the next page.

"NLIN" For printer terminals only: The format is output from the start of the next line (newline). No page feed takes place.

"PSTN" Only program-accessible fields are redisplayed on the screen, not the blanks.
PSTN presupposes that the format is already on the screen.

"ONLY" Only the data fields which do not have LOW-VALUE as the contents of their data transfer area are displayed. If, in addition, MAP-USE-ALL-ATTRS="N" (default) and FHS-MODY-ATTRS="Y", only the attribute fields of the data fields displayed are evaluated (see also the descriptions of the FHS-MODY-ATTRS and MAP-USE-ALL-ATTRS items).
In partial formatting any partial format can be formatted with FHS-MAPPING-METHOD="ONLY" even if the format has not been output before. FHS then automatically formats using FHS-MAPPING-METHOD="BEGN".

"RSET" The last format to have been output is then "reset" and output again, i.e.

- the contents of the output data transfer area are ignored
- protected fields on the screen are retained
- unprotected fields are filled with NULL characters
- detected fields may be detected again
- the cursor is set to the first unprotected or detectable field on the screen (on the 3270: to the first unprotected field).

"RSON" has the combined effect of RSET and ONLY.

Note

If FHS-MAPPING-METHOD has no value or one that is invalid, BEGN is assumed.

FHS-MODY-ATTRS

This item is used by the application program to determine whether during input/output formatting attributes are to be taken from the format definition or from the data transfer area. It is ignored in the case of #formats.
Meaning of the entries:

“Y” The attribute fields of the data transfer area are evaluated, i.e. it is possible to update attributes.

CAUTION!

FHS accepts any value in an attribute field which is not LOW-VALUE. Whenever FHS-MODY-ATTRS=“Y”, the user should insure that the contents of the attribute field are valid.

For any value other than “Y”, only the attributes from the format definition are used.

FHS-PARTIAL-MAP-OPT

This item is used by the application program to determine whether or not the format is to be used as a partial format.
Meaning of the entries:

”Y” The format is used as a partial format if it was defined as such (specified in IFG). FHS uses a special work area for partial formatting, the MAPLIST area; see page 395ff.

For any value other than “Y”, the format is formatted normally. If it was defined as a partial format, the start line is taken into account but a return code output.

FHS-MAP-PART

This item specifies the type of partial formatting call within a partial formatting cycle and is only taken into account for partial formatting (FHS-PARTIAL-MAP-OPT=“Y”); see also page 395ff.

Meaning of the entries:

“S” The call is not the last partial formatting call within a partial formatting cycle for output formatting.

“L” The call is the last or only partial formatting call within a partial formatting cycle for output formatting. The “WRTRD” call causes the input formatting of the first partial format to take place simultaneously.

”N” Only required for TIAM: FHS carries out input formatting for the next partial format for which FHS has received data. The CALL “WRTRD” call with FHS-MAP-PART=“N” does not initiate any input or output but merely the formatting of as yet unformatted partial formats read in with FHS-MAP-

PART="L". The name of the partial format is supplied in FHS-MAP-NAME. Further details on the input formatting of partial formats are given in the section starting on page 395.

For all values other than "S" and "N", "L" is assumed.

FHS-MAP-PART has no significance for serial input formatting with DCAM.

FHS-MAP-CURSOR-OPT

This item determines in output formatting whether the cursor is to be positioned explicitly using FHSCURS. It is ignored in the case of #formats. Meaning of the entries:

"Y" During output formatting the cursor is positioned in the field specified in the (previously called) CALL "FHSCURS".

For any value other than "Y", the cursor is positioned on the field specified when the format was defined.

FHS-SERVICE-FUNCTION

This item determines which service functions are to be executed with CALL "FHSSERV".

"1" The function 'Initialize data transfer area' is performed for #formats. In this, all field attributes are set in accordance with their default values in the format. The field contents and the global attributes (apart from 'Formatting acknowledgments') remain unchanged. It is thus possible to reset data transfer areas that have already been supplied with data to the initial status.

"2" The function "Determine name of character set" is performed. The format specified in FHS-MAP-NAME is thus loaded and the name of the corresponding character set is entered in the FHS-CCSN-INFO field; FHS-CCSN-INFO is contained in the data structure FHS-CCSN-PAR.

"3" The 'Unload Format' function is executed. The format specified in FHS-MAP-NAME is unloaded and can be replaced by a modified format.

"5" The 'Dynamic Retrieval of Information on the Structure of the Addressing Aid for #Formats' function is executed. You will find further information on this function in the MUCBL macro description under SERVICE=INFAREAS on page 254.

FHS-RESTART-OPT1

This item determines whether FHS should maintain a restart area during formatting to enable the contents of a screen to be restored.
Meaning of the entries:

"Y" A restart area is to be maintained.

For any value other than "Y", FHS does not maintain a restart area.

FHS-RESTART-OPT2

This item determines whether a restart should be performed for this call. FHS-RESTART-OPT2 is evaluated only when FHS-RESTART-OPT1="Y". Meaning of the entries:

"Y" Restart is performed.

For any value other than "Y", no restart is performed.

Notes on DCAM programs using the restart function

A terminal-specific administrative area to be specified in CALL "YSEND" and CALL "YRECEIVE" must be defined in DCAM-COBOL programs by the application program if restart areas have to be maintained.
This area must begin on a word boundary and the first two bytes must contain its length; the remainder of the area must be deleted with LOW-VALUE. There after the area no longer be accessed by the application program. FHS requires the area as a restart area and, if necessary, for the administration of partial formats. The following minimum lengths apply to the area (for a restart area of 2 Kbytes):

For a restart without partial formatting: 2066 bytes
For a restart with partial formatting: 3200 bytes

FHS checks whether the specified length of the administrative area is sufficient, depending on the required functions (restart and/or partial formatting). In the event of an error FHS supplies a return code.
This administrative area is defined e.g. as follows:

```
01      additional-area.
    41  length-field    PIC 9(4) COMP SYNC VALUE 2068.
    41  restart-area    PIC X(2066).
```

FHS-MAP-LIB-OPT

This item determines whether the subsequent FHS-MAP-LIB-NAME item is to be evaluated (optional).

Meaning of the entries:

"Y" The FHS-MAP-LIB-NAME item is evaluated. The format is taken from the format application file specified in this item.

For any value other than "Y", the FHS-MAP-LIB-NAME item is not evaluated.

A FILE command can be used prior to program start to specify a format application file. The FILE command has the format

```
/SET-FILE-LINK LINK=MAPLIB,FILE-NAME=filename
```

where filename is the name of the format application file.

The standard assignment at program start is the format application file F.MAPLIB.
The FILE command before the start of the program changes the standard assignment. It is, however, only evaluated once at the first CALL “FHSINIT” or first COBOL call for input/output with formatting (WROUT, WRTRD, YSEND or YRECEIVE) but has priority over the specification FHS-MAP-LIB-OPT=“Y”.
The assignment of a format application file remains effective until a new format application file entered in FHS-MAP-LIB-NAME is requested by means of the specification FHS-MAP-LIB-OPT=“Y” in a subsequent call (or the first one if no FILE command was entered) of the types mentioned above.

FHS-MAP-LIB-NAME

This item is where the application program writes the name of the format application file.
This item is evaluated only when FHS-MAP-LIN-OPT=“Y”. Ordinarily FHS expects to find the formats in the format application file F.MAPLIB. By changing the contents of this item it is possible to use more than one format application file. However, when a different format application file is used, the corresponding name must be entered and FHS-MAP-LIB-OBT=“Y” must be set. The format application file remains assigned until changed by means of FHS-MAP-LIB-NAME.

FHS-EXIT-LIB-OPT

This item determines whether the subsequent FHS-EXIT-LIB-NAME item is to be evaluated.
Meaning of the entries:

“Y” The subsequent FHS-EXIT-LIB-NAME item is evaluated. This item can be used to enter the name of a module library containing exit routines.

For any value other than “Y”, the FHS-EXIT-LIB-NAME item is not evaluated. When an exit routine is used, FHS ordinarily expects to find it in the F.EXITLIB module library.

FHS-EXIT-LIB-NAME

This item is where the application program writes the name of the module library containing the program’s exit routines.

Ordinarily, FHS expects to find the exit routines in the module library F.EXITLIB.

FHS-EXIT-FOR-OUTPUT

This item controls whether an exit routine is to be run during output formatting. It is ignored with #formats.
Meaning of the entries:

"Y" During output formatting an exit routine is run for each item for which the routine was specified at format definition time.

For any value other than "Y", no exit routine is initiated during output formatting.

FHS-EXIT-FOR-INPUT

This item controls whether an exit routine is run during input formatting. It is ignored with #formats.
Meaning of the entries:

"Y" During input formatting an exit routine is run for each item for which the routine was specified at format definition time.

For any value other than "Y", no exit routine is started during input formatting.

FHS-DESIRED-CCSNAME

This item is used to specify a user-defined CCS name if a 7-bit format is to be handled by FHS on an 8-bit terminal in exactly the same way as an 8-bit format. If a CCS name is specified in the control block, that format is used.

FHS-MAP-PAR-OPTIONAL

This part of FHS-MAP-PAR can be used to make optional specifications for format application.

If nothing is entered here, the following rules apply:

– For #formats, all specifications other than MAP-DEVICE-CLASS and MAP-PRINTER-CONTROL are ignored. It is advisable to supply the entire FHS-MAP-PAR-OPTIONAL with LOW VALUE. Differing inputs for MAP-DEVICE-CLASS and MAP-PRINTER-CONTROL are required only in the case of printers.

– If you call the FHSINIT subprogram in your application program (CALL "FHSINIT"), FHS alters the default values given in the FHS-INIT-PAR data structure. These new default values remain effective until the next FHSINIT call. FHS-MAP-PAR-OPTIONAL corresponds to the FHS-MAPPING-DEFAULTS portion of the FHS-INIT-PAR data structure.

– If you do not call the FHSINIT subprogram, default values are assumed. These appear underlined in the description of the data items.

When invalid entries are specified, FHS likewise assumes these default values or issues a return code.

MAP-DEVICE-CLASS

This item determines whether the format is to be output on a printer or a data display terminal.
The following entries are possible:

”DSS” Display terminal (8160, 9749, 975x, 9763, 3270)
At formatting time FHS uses the type of data display terminal it finds in the generation.

”DRS” Printer terminal. FHS distinguishes two types:

– Local printer attached to a data display terminal
A printer having the basic printer functions is assumed as the printer type. At formatting time FHS uses the data display terminal type to which the printer is connected as a secondary peripheral. FHS obtains the type of this control unit from the generation. If the special functions are to be used for output on the 90xx printers, the printer type must be specified explicitly.

– Central printer attached to a printer terminal controller
At formatting time FHS uses the type of printer terminal it finds in the generation.

“9001” The format is edited for the 9001 Printer. The special functions of this printer can be used.

“0131” The format is edited for the 9001-31 Printer. The special functions of this printer can be used.

“0189” The format is edited for the 9001-8931 Printer. The special functions of this printer can be used.

“9002” The format is edited for the 9002 Printer. The special functions of this printer can be used.

“9003” The format is edited for the 9003 Printer. The special functions of this printer can be used.

“9004” The format is edited for the 9004 Printer. The special functions of this printer can be used.

“1118” The format is edited for the 9011-18 Printer. The special functions of this printer can be used.

“1119” The format is edited for the 9011-19 Printer. The special functions of this printer can be used.

“9012” The format is edited for the 9012 Printer. The special functions of this printer can be used.

“9013” The format is edited for the 9013 Printer. The special functions of this printer can be used.

“9022” The format is edited for the 9022 Printer. The special functions of this printer can be used.

“PCL” The format is edited for the 9021 and 9022-200 Printers. The special functions of these printers can be used.

“3287” The format is edited for the 3287 Printer. The special functions of this printer can be used.

Note

The printer functions can be accessed via the MAP-PRINT-LINES, MAP-PRINT-COLUMNS, MAP-PRINT-PAPER and MAP-PRINT-FORM items in FHS-MAIN-PAR or FHS-INIT-PAR. Further printer functions can be accessed via the display attributes of the individual items which, in turn, can be changed in COBOL by means of attribute updating. Refer to the table in the Appendix for a list of which attributes are displayed and how for the individual printer types.

A 9014 must be generated as a 9013.

MAP-PRINTER-CONTROL

This item is evaluated only in the case of CALL “YSEND” (DCAM) for printer output.

“DRS” The printer is connected via a printer terminal controller.

“DSS” The printer is connected via a data display terminal.

MAP-AUTO-TAB

This item controls the tabulator function.

“Y” With automatic tabulator: The cursor jumps automatically from the end of an unprotected or detectable field to the beginning of the next unprotected or detectable field.

For the 3270 MAP-AUTO-TAB=“Y” always applies if the format was generated with IFG for application preparation for “fast formatting” and FHS-MODY-ATTRS=“N”; when FHS-MODY-ATTRS=“Y” this applies only to fields that are not accessible to the program and gaps between fields.

“N” Without automatic tabulator: The terminal user must position the cursor.

For the 3270 the field-specific ASKIP function can be implemented by means of the A-ASKIP attribute for fields that are accessible to the program. For other fields the tab is only controlled by the MAP-AUTO-TAB operand.

MAP-EFF-LEN

This item determines how the length fields in the data transfer area are to be supplied with information during input formatting for +formats.
The following entries are possible:

"Y" FHS enters the effective length of the input data in the length fields, if the associated field was updated.

"N" The contents of the length fields remain unchanged during input formatting.

"F" If the field was updated, the defined length of the field is entered in the length field.

Note

– Detectable (markable) fields (DET attributes) are always regarded as updated, except when FHS-MAPPING-METHOD="ONLY". The length field of a detectable field contains the value 0 when MAP-EFF-LEN="Y" and the defined field length when MAP-EFF-LEN="F".

– For IFG formats, FHS enters the total of the lengths of all fields within the group if MAP-EFF-LEN="F" and the sum of the "relevant" characters of the group fields within the group if MAP-EFF-LEN="Y". All characters with the exception of the declared fill character and the NULL character are "relevant" characters.

MAP-POS-DET-CHAR

In this item you can enter a printable character from the COBOL character set (with the exception of B) with which a detectable field detected at the data display terminal is filled during input formatting.

MAP-NEG-DET-CHAR

In this item you can enter a printable character from the COBOL character set (with the exception of B) with which a detectable field that has not been detected is filled during input formatting.

Note

– Wherever possible, both fill characters (MAP-POS-DET-CHAR and MAP-NEG-DET-CHAR) must be specified.

– The two fill characters must not be identical.

– If both items are specified incorrectly, the default values HIGH-VALUE (for "detected") and LOW-VALUE (for "undetected") are assumed.

MAP-READ-METHOD

This item determines the input mode of the data display terminal (READ MODIFIED or READ UNPROTECTED) for output formatting.
The following entries are possible:

“RUNP” “READ UNPROTECTED”: All unprotected fields and all protected fields with ‘automatic input’ (UNPROT or PROTRET attribute) are transmitted to the host computer. Fields which can be marked are not read, which means that FHS does not output these fields as markable. This attribute is suppressed during output formatting.

Note on the 3270

If FHS discovers during input formatting that a field for which input is expected is missing, it makes the following entries in the data transfer area: missing UNPROT fields are filled with input fill characters or NULL characters. When formatting is performed with a restart, missing PROTRET fields are supplied with the field contents of the restart area. When there is no restart area, nothing is entered in the PROTRET fields.

Update outputs (FHS-MAPPING-METHOD=“ONLY”, “RSET” or “RSON”) change the attributes of fields on the screen with regard to automatic input (PROTRET) if it is not possible to output them again using the update output, as follows:

– FHS-MAPPING-METHOD=“RSET”
All fields on the screen lose the automatic input attribute. PROTRET fields become PROT fields.

– FHS-MAPPING-METHOD=“ONLY” and
MAP-SCREEN-PRE-MODE=NO
All fields output again acquire the new attributes for automatic output requested; all fields not output again lose the auto-input attribute.

– FHS-MAPPING-METHOD=“ONLY” and
MAP-SCREEN-PRE-MODE=YES
All fields output again acquire the new attributes for automatic output requested; all fields not output again retain the auto-input attribute.

– FHS-MAPPING-METHOD=“RSON”
All fields output again acquire the new attributes for automatic input; the other fields lose the auto-input attribute.

"RMOD" READ MODIFIED": All fields modified at the terminal, all detectable fields and all fields with the FSET or PROTRET attributes are returned to the host computer and transmitted to the data transfer area. For all the remaining fields, the contents of the input data transfer area remain unchanged.

Note on the 3270

When formatting is performed with a restart, missing PROTRET fields are supplied with the field contents of the restart area. When there is no restart area, nothing is entered in the PROTRET fields.

Update outputs (FHS-MAPPING-METHOD="ONLY", "RSET" or "RSON") change the attributes of fields on the screen with regard to automatic input (PROTRET or FSET) if it is not possible to output them again using the update output, as follows:

– FHS-MAPPING-METHOD="RSET"
All fields on the screen lose the automatic input attribute. PROTRET fields become PROT fields.

– FHS-MAPPING-METHOD="ONLY" and
MAP-SCREEN-PRE-MODE=NO
All fields output again acquire the new attributes for automatic output requested; all fields not output again lose the automatic input attribute.

– FHS-MAPPING-METHOD="ONLY" and
MAP-SCREEN-PRE-MODE=YES
All fields output again acquire the new attributes for automatic output requested; all fields not output again retain the automatic input attribute.

– FHS-MAPPING-METHOD="RSON"
All fields output again acquire the new attributes for automatic input; the other fields lose the automatic input attribute.

MAP-SCREEN-PRE-MOD

This item is only evaluated for Data Display Terminals for output formatting when FHS-MAPPING-METHOD="ONLY".
Meaning of the entries:

"N" All fields updated or detected during the preceding input are reset to "not updated" or "undetected" during output.

"Y" All fields updated or detected during the preceding input retain this status.

MAP-READ-NILS

This item controls the handling of NULL characters in input fields.
The following entries are possible:

"Y" Reading with any NULL characters. FHS receives the contents of fields from the data display terminal during input transfer in the correct relative positions.

Exception

For the 3270 display terminal the effect is the same as MAP-READ-NILS="N" (governed by device characteristics).

The normal rules for field alignment apply.

"N" Reading without NULL characters. The data display terminal itself strips the NULL characters during the input transfer. The remaining characters are shifted to the left (implicit left alignment).

MAP-USE-ALL-ATTRS

This item controls the evaluation of the attribute fields in the data transfer area for output formatting when FHS-MODY-ATTRS= "Y" and FHS-MAPPING-METHOD="ONLY" or "RSON". In all other cases, the entry is ignored.
The following entries are possible:

"N" Only attribute fields whose associated data items do not contain LOW-VALUE are evaluated.

"Y" All the attribute fields are evaluated.

MAP-PRINTER-OPTION

This item can be used by the application program to determine whether a logical print acknowledgment is to be issued during output formatting operations at a printer terminal (printer acknowledgment).

– The acknowledgment is passed to the application program if the printer is connected centrally via a printer terminal controller.

– The acknowledgment is passed to the application program when the associated data display terminal is operating in bypass mode, i.e. when the format is output directly to the printer without being displayed at the terminal.
The following entries are possible:

"N" No acknowledgment is required.

"Y" FHS requires a logical print acknowledgment (positive or negative).

“E” Only a negative acknowledgment is required (if errors occurred during format output on a printer).

The printer acknowledgment can be read and formatted like a normal message. FHS informs the user in the item IN-PRINTER-RETURN-MSG whether the read message is a printer acknowledgment or a normal input message. The printer message contents are stored in the items PRINTER-MSG-TYPE, RETURN-BYTE1, RETURN-BYTE2 and RETURN-STATUS-INFO.

In each case given above, if MAP-PRINTER-OPTION is given the value “Y” or “E” and in addition, the items “MAP-PRINTER-RETURN-BYTE1” and “MAP-PRINTER-RETURN-BYTE2” are assigned any character from the COBOL character set, the contents of both printer return bytes are issued as an acknowledgment in the items RETURN-BYTE1 and RETURN-BYTE2.

Notes on requesting printer acknowledgments

1. Printer output with TIAM operating in bypass mode
   - If no acknowledgments are requested (MAP-PRINTER-OPTION=“N”), the format must be output using CALL “WROUT”.
   - If positive and negative acknowledgments are requested (MAP-PRINTER-OPTION=“Y”), the format must be output using CALL “WRTRD”.
   - Negative acknowledgments only (MAP-PRINTER-OPTION=“E”) are **not** permitted for output with TIAM operating in bypass mode.
2. Printer output with TIAM operating in hardcopy mode

   It is recommended that negative acknowledgments only or no acknowledgments be requested (MAP-PRINTER-OPTION=“N” or “E”), since each acknowledgment is an input for the WRTRD call by means of which the format is output.
3. Printer output with DCAM

   The DCAM application must manage all input messages and thus all acknowledgments itself. Restrictions such as those for TIAM do not apply to DCAM.

   In central hardcopy mode, no negative acknowledgments are output if only one printer is connected to the printer terminal controller.

MAP-PRINTER-RETURN-BYTE1
MAP-PRINTER-RETURN-BYTE2

These items are supplied with a character from the COBOL character set by the application program if acknowledgments are desired for output formatting on a printer. The user receives the contents of these bytes whenever MAP-PRINTER-OPTION is either “Y” or “E” (see page 323).

MAP-HARDCOPY-OPTION

This item controls message output on hardcopy units.

In addition, you specify whether the hardcopy device is locally or centrally connected (valid for output formatting only).
For the 3270 display terminal MAP-HARDCOPY-OPTION only controls the meaning of MAP-AUTO-HARDCOPY.
The following entries are possible:

"N" No full support for hardcopy units The cursor can be positioned in a protected field only when MAP-AUTO-TAB="N".
Any printable data item can be output on hardcopy devices.
MAP-AUTO-HARDCOPY is not evaluated for the 3270.

"L" Local hardcopy support is requested (a printer is connected locally via a data display terminal).
MAP-AUTO-HARDCOPY is evaluated for the 3270.

"C" Central hardcopy support is requested (output on a printer via a printer terminal controller). FHS takes the central address of the printer connected to the controller from the terminal characteristics.
MAP-AUTO-HARDCOPY is evaluated for the 3270.

Note

When MAP-HARDCOPY-OPTION = "L" or "C", the entire screen can be output on a hardcopy unit provided that the first data field is updatable or MAP-AUTO-TAB = "N". MAP-AUTO-TAB="N" may be omitted for the 8160 and 975x Data Display Terminals, as here the cursor may be located within protected fields.

MAP-AUTO-HARDCOPY

This item indicates whether or not automatic hardcopy mode is requested.
This item is ignored if MAP-HARDCOPY-OPTION="N".
The following entries are possible:

"N" Without automatic hardcopy mode.
Only manual hardcopy mode is supported. The terminal user must position the cursor himself and operate the LA1 key. The normal rules for cursor positioning following output apply.

"Y" Automatic hardcopy mode.
The entire message is automatically output on the hardcopy device. The normal rules for cursor positioning do not apply. The cursor is set to the first position on the screen once the output has terminated.

MAP-LOCK-KEYS

This item defines the keyboard status for Data Display Terminals.
The following entries are possible:

"N" The keyboard is not locked.

"Y" The keyboard is locked.

MAP-CLEAR-OPTION

This item is used by the application program for partial formatting to determine whether the screen is to be reconstructed. This item is only evaluated during partial formatting (FHS-PARTIAL-MAP-OPT="Y").
Meaning of the entries:

"N" The screen is not to be reconstructed.

"Y" The screen is to be reconstructed. At the same time it is also possible to define a new device type (see also page 395). "Y" may only be specified for the first partial format in the output cycle.

MAP-BEL-OPTION

This item controls the function "Alarm on Output".
Meaning of the entries:

"N" No alarm is triggered.

"Y" When a format is output on a data display terminal, visual (BEL indicator) and audible (short beep) alarms are triggered; only on Data Display Terminals with a special device option.
On the 3270 and printers, only an audible alarm.

MAP-PRINT-FORMAT-OPTION

In this section of the data structure FHS-MAIN-PAR, the application program can request special functions for output on a printer. Additional printer functions can be addressed via the display attributes of the individual fields.
For a list of how various attributes are represented for the individual printer types refer to the table in the Appendix.

MAP-PRINT-LINES

This item controls the line spacing of the printer for the format to be printed.
Meaning of the entries:

"N" Normal line spacing: 1/6 inch ≙ 4.23 mm

"S" Close line spacing: 1/8 inch ≙ 3.17 mm

"V" Minimum line spacing: 1/12 inch ≙ 2.12 mm

MAP-PRINT-COLUMNS

This item controls the character spacing of the printer for the format to be printed.
Meaning of the entries:

"N" Normal character spacing: 1/10 inch ≙ 2.54 mm (normal font)

"S" Close character spacing: 1/12 inch ≙ 2.12 mm (condensed font 1)

"V" Minimum character spacing:
1/17 inch ≙ 1.49mm for 9001
1 /15 inch ≙ 1.69 mm for 90xx, except 9001
(condensed font 2)

The table on page 536 shows the maximum number of characters per line for each printer type.

MAP-PRINT-PAPER

This item selects the type of paper feed for printers. For all other terminals or for printers without a single-sheet feed the entry is ignored.
Meaning of the entries:

"0" The format is intended for continuous paper feed.

"1" This format is intended for single-sheet feed from cartridge 1; after printing the sheet is not ejected.

"2" This format is intended for single-sheet feed from cartridge 2; after printing the sheet is not ejected.

"3" This format is intended for single-sheet feed from cartridge 3; after printing the sheet is not ejected.

"9" This format is intended for the form feed attachment on the 9013 Printer; after printing the sheet is not ejected.

"A" This format is intended for single-sheet feed from cartridge 1; after printing the sheet is ejected.

"B" This format is intended for single-sheet feed from cartridge 2; after printing the sheet is ejected.

"C" This format is intended for single-sheet feed from cartridge 3; after printing the sheet is ejected.

"I" This format is intended for the form feed attachment on the 9013 Printer; after printing the sheet is ejected.

Note

When a form feed attachment is used on the 9013 Printer, the previous sheet must be explicitly ejected before a new sheet is inserted.

MAP-PRINT-FORM

This item is used by the application program to tell FHS how the paper is inserted in the cartridge or form feed attachment. The entry is only evaluated for output using single-sheet feed on a printer (MAP-PRINT-PAPER="1", "2", "3", "A", "B" or "C") or output using the form feed attachment on a 9013 Printer (MAP-PRINT-PAPER="9" or "I"). The entry presupposes that paper has been correctly inserted in the cartridge or form feed attachment.
Meaning of the entries:

"L" Paper inserted for single-sheet feed or formfeed attachment in portrait format.

"B" Paper inserted for single-sheet feed or form feed attachment in landscape format.

Note

The data fields MAP-LIB-LOAD-OPTION, MAP-LIB-LOAD-MODE and MAP-LIB-LOAD-FILE are provided in the data structure only for reasons of compatibility.

MAP-HOLE-COLOR

This field is used to specify the color for the spaces between fields.

"U" Spaces between fields are output as protected fields which cannot be transferred. On Data Display Terminals with default settings, these spaces are displayed with reduced intensity.

"W" Spaces between fields are displayed with normal intensity.

"G" Spaces between fields are displayed with reduced intensity.

This field is ignored for formats with 'fast formatting'. It is meaningful only for Data Display Terminals which support the function 'reduced intensity' (9755, 9763).

FHS-EXIT-PAR

This part of the FHS-MAIN-PAR data structure corresponds to the user exit interface. It is required only when an exit routine is used.

EXIT-IDENT-LEN

This item contains the maximum length of the exit remark for the exit routine (=8 for IFG formats).

EXIT-IDENT

This item contains the exit remark for the exit routine (IFG entry).

EXIT-IN-OUT

This item contains

“I” for input formatting

“O” for output formatting.

EXIT-RET-INFO

This item is used for exchanging information between exit routine and main program, e.g. the exit routine can use it to enter a return code which is evaluated by the application program.

EXIT-U-FLAG

This field contains the type of field currently processed in the exit-routine.
If the value is 'U', then the field is a UNICODE field.
If the value is a space, then the field is a 7-/8-bit field (defined only if FHSMAINU is used).

EXIT-FLD-LEN

This item contains the defined length of the field that was passed to the exit routine.

EXIT-EFF-LEN

This item contains the effective length of the data in the field that was passed to the exit routine.

EXIT-DATA-U

This item contains the contents of the field in the form of a Unicode string of characters (defined only if FHSMAINU is used).

EXIT-DATA

This item contains the contents of the field that was passed to the exit routine.

Note

The data structure FHS-EXIT-PAR does not contain any items for the attribute fields. If you also want to evaluate the attribute fields in an exit routine, the routine must be coded in ASSEMBLER (see page 289ff).

7.2.2 The FHS-INIT-PAR data structure

FHS-INIT-PAR is the name of the following data structure, which is copied into the application program by means of **COPY “FHSINITP”.** It is required for the FHS COBOL call “FHSINIT”, which is used to initiate formatting and to define formatting default values.

```
01  init-area.
    COPY FHSINITP.
*****************************************************************
* NAME              FHSINITP                                    *
* VERSION           811                                         *
*                                                               *
*    Data structure for FHSINIT call                            *
*                                                               *
* END-INTERFACE     FHSINITP                                    *
*****************************************************************
 35  FHS-INIT-PAR.
*
     41  FHS-INIT-PAR-GENERAL.
         42  FHS-I-O-AREA-LEN         PIC 9(5) COMP SYNC.
         42  FHS-RES-MAP-NO           PIC 9(4) COMP SYNC.
         42  FHS-MAP-NO               PIC 9(4) COMP SYNC.
         42  FILLER                   PIC X(7).
         42  FHS-ACCESS-METHOD        PIC X.
*
     41  FHS-MAPPING-DEFAULTS         PIC X(60).
*
     41  FHS-MAP-OPTIONS   REDEFINES  FHS-MAPPING-DEFAULTS.
         42  MAP-DEVICE-TYPE          PIC X(4).
         42  MAP-CONTROL-UNIT         PIC X(4).
         42  MAP-USER-AREA-LEN        PIC 9(5) COMP SYNC.
         42  MAP-AUTO-TAB             PIC X.
         42  MAP-EFF-LEN              PIC X.
         42  MAP-POS-DET-CHAR         PIC X.
         42  MAP-NEG-DET-CHAR         PIC X.
         42  FILLER                   PIC X(8).
         42  MAP-READ-METHOD          PIC X(4).
         42  MAP-SCREEN-PRE-MOD       PIC X.
         42  MAP-READ-NILS            PIC X.
         42  MAP-USE-ALL-ATTRS        PIC X.
         42  MAP-PRINTER-OPTION       PIC X.
         42  MAP-PRINTER-RETURN-BYTE1 PIC X.
         42  MAP-PRINTER-RETURN-BYTE2 PIC X.
         42  FILLER                   PIC X.
         42  MAP-HARDCOPY.
             43  HARDCOPY-OPTION      PIC X.
             43  CENTRAL-PRINT-ADDR   PIC 9(2) COMP SYNC.
         42  MAP-AUTO-HARDCOPY        PIC X.
```

```
          42  MAP-LOCK-KEYS            PIC X.
          42  MAP-CLEAR-OPTION         PIC X.
          42  MAP-BEL-OPTION           PIC X.
          42  MAP-PRINT-FORMAT-OPTION.
              43  MAP-PRINT-LINES      PIC X.
              43  MAP-PRINT-COLUMNS    PIC X.
              43  MAP-PRINT-PAPER      PIC X.
              43  MAP-PRINT-FORM       PIC X.
          42  MAP-LIB-LOAD-OPTION.
              43  MAP-LIB-LOAD-MODE    PIC X.
              43  MAP-LIB-LOAD-FILE    PIC X.
          42  MAP-HOLE-COLOR           PIC X.
          42  FILLER                   PIC X(11).
*
      41  FHS-INIT-SYS-INFO            PIC X(80).
*
      41  FHS-BS2000-INFO   REDEFINES  FHS-INIT-SYS-INFO.
          42  FILLER                   PIC X(25).
          42  FHS-MAP-LIB-OPT          PIC X.
           42  FHS-MAP-LIB-NAME         PIC X(54).
```

Description of the data items

Data items not described below are identical to data items in the data structure FHS-MAIN-PAR and are described only there.

FHS-INIT-PAR-GENERAL

This part of the FHS-INIT-PAR data structure is used for defining general entries for the FHSINIT call. The FHS-INIT-PAR-GENERAL area is evaluated only when first called with CALL “FHSINIT”. If the “FHSINIT” call is specified several times, the contents of FHS-INIT-PAR-GENERAL are **not** evaluated for the remaining calls since the formatting was initiated with the first FHSINIT call.

FHS-I-O-AREA-LEN

This item is **not** evaluated by FHS and does not need to be supplied. It is retained for compatibility only.

FHS-RES-MAP-NO

This item is used for specifying the number of formats (100 max.) to be loaded on opening formatting.
If FHS-RES-MAP-NO ≠ 0, the name of a list containing the names of the formats to be loaded on opening formatting must be entered in the FHSINIT call under USING as the fourth operand. If the value of FHS-RES-MAP-NO is n, the first n formats from this format list are loaded as soon as formatting is commenced. The format list must not contain fewer elements than specified in FHS-RES-MAP-NO. Also, the value of FHS-RES-MAP-NO must not exceed the value of FHS-MAP-NO.

FHS-MAP-NO

This item controls the size of the directory for format definitions in main memory. All values between 0 and 2730 are valid for FHS-MAP-NO; if the item has 0 or no value at all assigned to it, FHS assumes the default value 100. The value of FHS-MAP-NO must be greater than the total number of **all** formats (including partial formats and character set formats) used and at least as large as the value of FHS-RES-MAP-NO.

Note

The FHS COBOL module MFHSCALL contains an area for the directory in which all formats used are entered. As the default option, up to 100 entries are possible.

If the value of FHS-MAP-NO is >100 but <2730, MFHSCALL attempts to obtain storage space for a larger directory. If no more memory is available, FHS issues a warning (FHS-MAIN-RC=8, ERROR-CATEGORY=80, ERROR-REASON=8) and uses the internal directory for max. 100 entries. FHS-MAP-NO is reset to the default value of 100. If the value of FHS-RES-MAP-NO is greater than 100, this value is also reset internally to 100 and only the first 100 formats from the format list are loaded in response to CALL “FHSINIT”.

FHS-MAPPING-DEFAULTS

This part of FHS-INIT-PAR is used for entering optional specifications for format application. Here the user defines his own default values. If no entries are made here, the standard default values apply. The user may define or modify his default values with each formatting (see FHS-MAIN-PAR).

– If CALL “FHSINIT” is not called, default values are assumed. These values are underlined in the description of the data items.

– If the entry is omitted or invalid, FHS also assumes these default values or issues a return code. If the FHSINIT subprogram is called several times, the underlined default values are no longer valid for further calls; the default values are those specified with the previous FHSINIT call. For this reason, all required modifications must be explicitly specified.

MAP-USER-AREA-LEN

This item indicates the maximum length of the data transfer area for input/output (optional). You can specify any value between 1 and 32767. If the specified value is exceeded during input formatting, the formatting is aborted. In output formatting the data is taken from the data transfer area if it does not exceed the specified length. If it does, the fields in the format accessible to the application program are filled with the fill character or NULL character (if no fill character was defined).

CENTRAL-PRINT-ADDR

This item is used to give the channel address of the printer to the printer terminal controller (central printer address) when HARDCOPY-OPTION=“C”. FHS updates this address at formatting time.

7.2.3 The FHS-ATTR-PAR data structure

This section describes field attribute updating for formats that do **not** use the data transfer area with separate attribute blocks and field contents For #formats, this section is of significance only when using the field attribute group ‘Attribute Combination’ (see page 62ff).

FHS-ATTR-PAR is the name of the data structure below, which is copied into the application program by means of **COPY “FHSATTRP”.** It is required whenever field attributes are updated using the FHSATTR call.

A table showing the display characteristics on the individual data display terminals and printers is given in the Appendix.

```
01 attr-area.
   COPY FHSATTRP.
*****************************************************************
* NAME              FHSATTRP                                    *
* VERSION           811                                         *
*                                                               *
*    Data structure for FHSATTR call                            *
*                                                               *
* END-INTERFACE     FHSATTRP                                    *
*****************************************************************
 35  FHS-ATTR-PAR.
*
     41  FHS-ATTR-PAR-BASIC.
         42  A-UPDATE-METHOD          PIC X(3)  VALUE „REP“.
         42  FILLER                   PIC X(5).
         42  A-PROT-LEVEL             PIC X(4).
         42  A-DISP-LEVEL             PIC X.
         42  FILLER                   PIC X(3).
*
     41  FHS-ATTR-PAR-OPTIONAL        PIC X(24).
*
     41  FHS-ATTR-OPTIONS  REDEFINES  FHS-ATTR-PAR-OPTIONAL.
         42  A-NO-HARDCOPY            PIC X.
         42  A-NUMERIC                PIC X.
         42  A-SIGNAL                 PIC X.
         42  A-ITALIC                 PIC X.
         42  FILLER                   PIC X(12).
         42  A-WIDE                   PIC X.
         42  A-TALL                   PIC X.
         42  FILLER                   PIC X(5).
         42  A-ASKIP                  PIC X.
```

Description of the data items

FHS-ATTR-PAR-BASIC

This part of the FHS-ATTR-PAR data structure is used to indicate the type of function to be executed as well as the most important attributes (e.g. protected/unprotected).

A-UPDATE-METHOD

This item indicates the type of function to be executed. Currently only the “REP” (REPLACE) function is supported. Meaning of the entries:

“REP” The previous contents of the attribute field are replaced. Attributes for which nothing was entered in FHS-ATTR-PAR are given default values. All the attributes of a field should be specified in FHSATTR; it is not enough to enter only the ones being updated.

Any value not equal to “REP” causes the FHSATTR call to be aborted, i.e. the attribute field is not modified.

A-PROT-LEVEL

This item is used to specify whether a field is protected, unprotected or detectable, and whether or not it is to be transferred during input. Meaning of the entries:

“UNPR” The field is unprotected. It can be overwritten on the terminal. It is returned automatically after being input when MAP-READ-METHOD=“RUNP” (READ UNPROTECTED), or after being detected when MAP-READ-METHOD=“RMOD” (READ MODIFIED).

“PROT” The field is protected. It cannot be overwritten on the terminal and is not returned after being input.

“PRET” The field is protected. It cannot be overwritten on the terminal and is always returned after being input.

“FSET” The field is unprotected. It can be overwritten on the terminal and is returned after being input. In the case of “READ UNPROTECTED” (MAP-READ-METHOD=“RUNP”), this attribute is treated in the same manner as “UNPR”.

“PDET” The field is protected and detectable.

Note on formats for the 3270 display terminal

Fields with the DET attribute and fields with the BRT attribute without DET are only detectable if the first character is a designator character.

Designator characters for

– selection fields:
’?’ becomes ’>’ when selected and is reset to ’?’ when selected again. The selection does not immediately trigger input.

– attention fields:
Type 1 designator characters are NULL or blank. Selection triggers immediate input (field addresses of all modified fields only). All fields with the attribute BRT and NULL or blank as the first character are type 1 attention fields. However, FHS does not permit selection of such a field as data loss may occur; FHS provides a return code. The type 2 designator character is ’&’. Selection triggers immediate input (field addresses and field contents of all modified fields).

You can enter designator characters for DET fields in the data transfer area prior to output formatting yourself. Note, however, that FHS treats the designator characters like regular field contents as regards justification and fill characters, i.e.

– in right justification the designator character is shifted to the right and loses its function

– in left justification the designator character disappears if it is the same as the fill character.

DET fields therefore undergo postprocessing by FHS for output. After the function just & fill and return from the exit routine for output, FHS checks the first character of the field contents. In the case of MAP-READ-METHOD=“RMOD” and with the field attribute DET, only ‘?’ or ‘&’ is permitted as the first character; other first characters are output as ‘?’ by FHS.

A-DISP-LEVEL

This item determines the brightness of the field on the data display terminal.

“B” The field is displayed at full brightness, or in green (or red in conjunction with “A-ITALIC”) on the 9752 Data Display Terminal.

“H” The field is displayed with normal brightness, or in yellow (or white in conjunction with “A-ITALIC”) on the 9752 Data Display Terminal.

“D” The field is invisible.
On the 3270 the field is also non-detectable and non-printable.

If nothing is entered here, FHS assumes the following default values:

“B” for the “UNPR” and “FSET” attributes,

“H” for the “PROT”, “PRET” and “PDET” attributes.

FHS-ATTR-PAR-OPTIONAL

This part of the data structure contains the remaining field attributes. The only attributes considered are those which have been given the value “Y”.

A-NO-HARDCOPY

If this item is given the value “Y”, the field cannot be printed on the hardcopy device. On the 3270 the field is also invisible and non-detectable.

A-NUMERIC

If this item is given the value “Y”, the field only accepts numeric data, i.e. the digits 0 through 9 and the characters “*”, “+”, “-”, “/”, “.” and “,”; on the 3270 only the digits 0 through 9, the characters “.” and “-”, and the DUP (duplicate) key.

During output, any characters may appear in the numeric fields.

This attribute must not be specified in conjunction with the “PROT”, “PRET” or “PDET” attribute.

On the 3270 the attribute is only evaluated for fields that are accessible to the program.

A-SIGNAL If this item is given the value “Y”, the field flashes when displayed on the data display terminal. This item is ignored if it occurs in conjunction with the “PDET” attribute. A-SIGNAL is not evaluated for the 3270.

A-ITALIC The field is italicized or, in the case of most data display terminals, underlined (see table on page 536ff).

A-WIDE If this item is given the value “Y”, the field is printed in double-width font on the printers.

On the some printers the wide type is simulated, i.e. a blank is inserted after each character (see table on page 536ff).

A-TALL If this item is given the value “Y”, the field is output in tall type on the some printers. On all other devices this entry is ignored (see table on page 536ff).

A-ASKIP Only for the 3270
If this item has the value “Y”, the field is automatically skipped by the cursor. A-ASKIP=“Y” is only evaluated when A-PROT-LEVEL=“PROT” or “PRET”, in which case A-NUMERIC is ignored. In combinations of ASKIP with A-PROT-LEVEL=“PDET”, “UNPR”, “FSET” and A-NUMERIC=“Y”, ASKIP is ignored. Field-specific ASKIP is only possible with MAP-AUTO-TAB=“N”.

7.2.4 The FHS-EXITMOD-PAR data structure

If the exit routine is *not* foreseen to handle UNICODE fields, the FHS-EXITMOD-PAR data structure must be copied into the exit routine by means of **COPY "FHSEXITP".** It is required in the linkage section of the exit routine, and corresponds to the user exit interface (see also the last part of the FHS-MAIN-PAR data structure).

```
 01  exit-area.
     COPY FHSEXITP.
*****************************************************************
*                                                               *
*    FHSEXITP Version 811                                       *
*                                                               *
*       DATA STRUCTURE FOR THE EXIT ROUTINE                     *
*                                                               *
*****************************************************************
 35  FHS-EXITMOD-PAR.
*
     41  EXITMOD-PAR                  PIC X(108).
*
     41  FHS-EXIT-PAR      REDEFINES  EXITMOD-PAR.
         42  EXIT-IDENT-LEN           PIC 9(5) COMP SYNC.
         42  EXIT-IDENT               PIC X(8).
         42  EXIT-IN-OUT              PIC X.
         42  EXIT-RET-INFO            PIC X.
         42  FILLER                   PIC XX.
         42  FILLER                   PIC X(4).
         42  EXIT-FLD-LEN             PIC 9(5) COMP SYNC.
         42  EXIT-EFF-LEN             PIC 9(5) COMP SYNC.
         42  EXIT-DATA                PIC X(80).
```

If the exit routine is foreseen to handle UNICODE fields, the FHS-EXIT-MOD-PAR-U data structure must be used by means of **COPY "FHSEXITU"**. This will generate the following data structure:

```
 01  exit-area.
     COPY FHSEXITU.
*****************************************************************
*                                                               *
* FHSEXITU Version 850                                          *
*                                                               *
* DATA STRUCTURE FOR THE EXIT ROUTINE                           *
*                                                               *
*****************************************************************
 35 FHS-EXITMOD-PAR.
*
    41 EXITMOD-PAR             PIC X(292).
*
    41 FHS-EXIT-PAR REDEFINES EXITMOD-PAR.
       42 EXIT-IDENT-LEN       PIC 9(5) COMP SYNC.
       42 EXIT-IDENT           PIC X(8).
       42 EXIT-IN-OUT          PIC X.
       42 EXIT-RET-INFO        PIC X.
       42 FILLER               PIC XX.
       42 FILLER               PIC X(2).
       42 EXIT-U-FLAG          PIC X.
       42 FILLER               PIC X.
       42 EXIT-FLD-LEN         PIC 9(5) COMP SYNC.
       42 EXIT-EFF-LEN         PIC 9(5) COMP SYNC.
       42 EXIT-DATA-U          PIC N(132).
       42 EXIT-DATA REDEFINES EXIT-DATA-U PIC X(264).
```

The end of the FHS-MAIN-PAR data structure is changed accordingly. If a program must handle UNICODE fields, then FHSMAINU must be used instead of FHSMAINP.

Description of the data items

FHS-EXIT-PAR

This data structure corresponds to the user exit interface. It is only required when an exit routine is used.

EXIT-IDENT-LEN

This item contains the maximum length of the exit remark for the exit routine (=8 for IFG formats).

EXIT-IDENT

This item contains the exit remark for the exit routine (IFG entry).

EXIT-IN-OUT

This item contains

“I” for input formatting

“O” for output formatting.

EXIT-RET-INFO

This item is used for exchanging information between exit routine and main program, e.g. the exit routine can use it to enter a return code which can be evaluated by the application program.

EXIT-U-FLAG

If this field contains the character 'U', the field content returned by FHS in the field EXIT-DATA-U must be handled by the application as a UNICODE field. Else, the text returned by FHS to the application program must be handled as “normal” text using the EXIT-DATA field (redefined on the EXIT-DATA-U field).

EXIT-FLD-LEN

This item contains the defined length of the field which was transferred to the exit routine.

EXIT-EFF-LEN

This item contains the effective length of the data in the field transferred to the exit routine.

EXIT-DATA

This item comprises the contents of the field transferred to the exit routine.

EXIT-DATA-U

This item comprises the contents of the field transferred to the exit routine, if the EXIT-U-FLAG contains the character ’U’.

Note

FHS-EXIT-PAR contains no items for the attribute fields. If attribute fields are to be evaluated in an exit routine, the routine must be coded in ASSEMBLER (see page 289ff).

7.2.5 The FHS-CCSN-PAR data structure

FHS-CCSN-PAR is the name of the data structure below, which is copied into the application program by means of **COPY “FHSCCSNP”.** It is required for the COBOL call CALL “FHSSERV” to determine the character set name of a format.

```
 01  ccsn-area.
     COPY FHSCCSNP.
*****************************************************************
*                                                               *
*    FHSCCSNP Version 811                                       *
*                                                               *
*       FHS-CCSN-PAR                                            *
*                                                               *
*  Parameter list used by FHSSERV to receive the CCSNAME        *
*  of a format.                                                 *
*                                                               *
*****************************************************************
 35  FHS-CCSN-PAR.
*
     41  FILLER                      PIC X(8).
     41  FHS-CCSN-INFO               PIC X(8).
     41  FILLER                      PIC X(16).
```

Description of the data items

FHS-CCSN-INFO

This item is where FHS writes the name of the character set after the FHSSERV call. Blanks are entered for 7-bit formats.

7.3 COBOL calls in the access methods for formatted input/output

The BS2000 access methods TIAM (for time sharing) and DCAM (for inquiry and transaction processing), together with their COBOL calls, enable the user to output and input a formatted screen. The formatting does not have to be called separately since it is integrated into the calls.

For each call, there is a diagram showing:

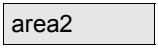
– which data structures the call uses,

– which areas must or may be filled prior to the call,

– which information is returned by FHS.

The different shadings in the diagrams have the following meaning:

This area must be filled by the application program before the call.



This area contains the information returned by FHS.

7.3.1 TIAM calls

FHS is integrated into the TIAM COBOL calls CALL **“WROUT”** and CALL **“WRTRD”.** These calls are described in the “TIAM (TRANSDATA, BS2000)” User Guide. The following paragraphs contain additional information on the use of FHS.

7.3.1.1 TIAM call for outputting formatted messages

```
CALL "WROUT" USING     TIAM-CONTROL-INFO
                       transfer-area
                       FHS-MAIN-PAR [(i)].
```

TIAM-CONTROL-INFO
controls the TIAM call.

transfer-area

2 is the name of the data transfer area. In the case of CALL “WROUT”, the data transfer areas - as provided (with addressing aids) by IFG - must be preceded by an item in which FHS writes the length of the message.
This can be done as follows:

```
01    transfer-area.
      03 length-field              PIC 9(5) COMP.
      03 format-name.
      COPY format-name.
```

FHS-MAIN-PAR
controls formatting. If the data structure is defined by means of tables and occurs several times, the appropriate index must also be specified.

Before the TIAM call CALL **“WROUT”** to output a formatted message is issued, the data transfer area and the items below must be supplied with the following information:

1 The **EDIT-MODE** item in TIAM-CONTROL-INFO **must** be given the value **“F”** (for “FHS”).

6 The **FHS-MAP-NAME** item in FHS-MAIN-PAR **must** contain the name of the format to be output.

In addition, the following optional entries in FHS-MAIN-PAR are possible:

9 If the formats are not located in the standard format application file (F.MAPLIB), the following items must also be filled:

- **FHS-MAP-LIB-OPT** with the value “**Y**” and
- **FHS-MAP-LIB-NAME** with the name of the format application file in which the desired format is stored as a module.

8 If the cursor is to be positioned explicitly and CALL “FHSCURS” has been called, the **FHS-MAP-CURSOR-OPT** item must be given the value “**Y**” (does not apply to #formats).

8 If field attributes are to be modified in +formats (with CALL “FHSATTR” or a MOVE statement), the **FHS-MODY-ATTR** item must be given the value “**Y**”.

8 If partial formats are to be used, the following items must also be filled:

- **FHS-PARTIAL-MAP-OPT** with the value “**Y**” and
- **FHS-MAP-PART** with the value “S” if it is not the last call in the partial formatting cycle or the value “L” if it is the last call in this cycle, and
- **MAP-CLEAR-OPT** with the value “Y” for the first call in a partial formatting cycle, and the value “N” for subsequent calls in this cycle (not for #formats).

7 When an exit routine is used, the following items must also be filled:

- **FHS-EXIT-MOD-NAME** with the name of the exit routine,
- **FHS-EXIT-LIB-OPT** with a “Y” if the exit routine is not in the standard library F.EXITLIB,
- **FHS-EXIT-LIB-NAME** with the name of the module library containing the exit routine (only if FHS-EXIT-LIB-OPT=“Y”),
- **FHS-EXIT-FOR-OUTPUT** with a “Y” if the exit routine is to be called during output formatting.

8 The remaining items in FHS-MAIN-PAR may be used to control further formatting characteristics (see page 303ff, FHS-MAIN-PAR).

FHS returns:

global and field attributes,

3 the primary return code in **FHS-MAIN-RC** (see page 524ff),

4 the secondary return code in **ERROR-CATEGORY** and **ERROR-REASON** (see page 524ff),

5 the minimum length computed for the data transfer area in **OUT-USER-AREA-LEN** and whether the data transfer area is large enough in **OUT-USER-AREA-TRUNCATION**.

These items are evaluated by the user in addition to the TIAM return code following a CALL “WROUT” for formatted messages. The value 0 indicates that the formatting was error-free.

CALL	USING	Length	Descrip-tion	Entry in data item
„WROUT“	TIAM-CONTROL-INFO			
	EDIT-OUT			
	EDIT-MODE	1	1.	„F“
	transfer-area		2.	
	FHS-MAIN-PAR			
	FHS-CONTROL-INFO			
	FHS-MAIN-RC	2	3.	primary return code
	FHS-ERROR-INFO			
	ERROR-CATEGORY	2	4.	secondary return code
	ERROR-REASON			
	...			
	OUT-USER-AREA-LEN	4	5.	length of transfer area
	OUT-USER-AREA-TRUNCATION	1	5.	LOW-VALUE or HIGH-VALUE
	...			
	FHS-MAP-PAR			
	FHS-MAP-NAME	8	6.	„name-of-format“
	FHS-EXIT-MODE-NAME	8	7.	„name-of-exit-routine“
	...		8.	formatting specifications
	FHS-MAP-LIB-OPT	1	9.	„Y“
	FHS-MAP-LIB-NAME	54	9.	„name-of-format-file“
	FHS-EXIT-LIB-OPT	1	7.	„Y“
	FHS-EXIT-LIB-NAME	54	7.	„file-for-exit-routine“
	FHS-EXIT-FOR-OUTPUT	1	7.	„Y“
	...			
	FHS-MAP-OPTIONS	60	8.	formatting specifications
	...			

FHS returns

TIAM CALL „WROUT“

7.3.1.2 TIAM call for the input and output of formatted messages

```
CALL "WRTRD" USING     TIAM-CONTROL-INFO
                       transfer-area-1
                       transfer-area-2
                       FHS-MAIN-PAR [(i)].
```

where

TIAM-CONTROL-INFO
controls the TIAM call.

transfer-area-1
2 is the name of the output data transfer area.

transfer-area-2
3 is the name of the input data transfer area.

In the case of CALL “WRTRD”, the data transfer areas as provided (with addressing aids) by IFG must be preceded by an item in which FHS writes the length of the message.
This can be done as follows:

```
01     transfer-area-1.
       03 length-field-1            PIC 9(5) COMP.
       03 format-name-O.
       COPY format-name-O.
     *
01     transfer-area-2.
       03 length-field-2            PIC 9(5) COMP.
       03 format-name-I.
       COPY format-name-I.
```

transfer-area-1 and transfer-area-2 may be identical if FHS is to use the same data transfer area for input and output. transfer-area-1 and transfer-area-2 must be identical for #formats.

FHS-MAIN-PAR

controls formatting. If the data structure is defined by means of tables and occurs several times, the appropriate index must also be specified.

Before the TIAM call CALL “**WRTRD”** for outputting or inputting a formatted message is issued, the data transfer area and the items below must be supplied with information as follows:

1 The **EDIT-MODE** item in TIAM-CONTROL-INFO **must** be given the value “**F”** (for “FHS”).

14 The **FHS-MAP-NAME** item in FHS-MAIN-PAR **must** contain the name of the format to be output.

In addition, the following optional entries in FHS-MAIN-PAR are possible:

17 If the formats are not located in the standard format application file (F.MAPLIB), the following items must also be filled:

– **FHS-MAP-LIB-OPT** with the value “**Y”**

– **FHS-MAP-LIB-NAME** with the name of the format application file in which the desired format is stored as a module.

16 If the cursor is to be positioned explicitly and CALL “FHSCURS” has been called, the **FHS-MAP-CURSOR-OPT** item must be given the value “**Y”** (does not apply to #formats).

16 If field attributes are to be modified in +formats (with CALL “FHSATTR” or a MOVE statement), the **FHS-MODY-ATTRS** item must be given the value “**Y”.**

16 If partial formats are to be used, the following items must also be filled:

– **FHS-PARTIAL-MAP-OPT** with the value “**Y”** and

– **FHS-MAP-PART** with

“S” if it is not the last call in this partial formatting cycle, and

“L” if it is the last CALL “WRTRD” in this cycle, and

“N” if merely an input formatting operation is to be performed for the next partial format, and

– **MAP-CLEAR-OPT** with the value “Y” for the first call in a partial formatting cycle for output formatting, and the value “N” for subsequent calls in this cycle.

15 When an exit routine is used, the following items must also be filled:

– **FHS-EXIT-MOD-NAME** with the name of the exit routine

– **FHS-EXIT-LIB-OPT** with a “Y” if the exit routine is not in the standard library F.EXITLIB,

– **FHS-EXIT-LIB-NAME** with the name of the module library containing the exit routine (only if FHS-EXIT-LIB-OPT=“Y”),

- **FHS-EXIT-FOR-OUTPUT** with a “Y” if the exit routine is to be called during output formatting,
- **FHS-EXIT-FOR-INPUT** with a “Y” if the exit routine is to be called during input formatting.

16 The remaining items in FHS-MAIN-RC may be used to control further formatting characteristics (see page 303ff, FHS-MAIN-PAR).

FHS returns:

global and field attributes,

4 the primary return code in **FHS-MAIN-RC** (see page 524ff); if it has the value 0, the formatting operation was error-free.

5 the secondary return code in **ERROR-CATEGORY** and **ERROR-REASON** (see page 524ff).

6 whether the input message in the item **IN-PRINTER-RETURN-MSG** was a printer acknowledgment. If so, FSH also fills the following items:

- **RETURN-MSG-TYPE** with the type of acknowledgment (+ve or -ve),
- **RETURN-BYTE1** and **RETURN-BYTE2** with the printer return bytes,
- **RETURN-STATUS-BYTE** with the contents of the status byte.

7 the minimum length computed for the output data transfer area in **OUT-USER-AREA-LEN** and whether the data transfer area is large enough in **OUT-USER-AREA-TRUNCATION**,

8 whether a field was detected during input formatting (in the **IN-FIELD-DET** item),

9 whether data was received (**IN-MSG-NILS** item),

10 whether a function key was activated (**IN-F-KEY** item),

11 whether a short message was received (**IN-K-KEY** item),

12 the length of the data transfer area (**IN-USER-AREA-LEN** item),

13 the length of the input data (**IN-MSG-LEN**).

These items are evaluated by the user in addition to the TIAM return code following a WRTRD call for formatted messages.

CALL	USING	Length	Descr.	Entry in data item
„WRTRD“	TIAM-CONTROL-INFO			
	EDIT-OUT			
	EDIT-MODE	1	1.	„F“
	transfer-area-1		2.	
	transfer-area-2		3.	
	FHS-MAIN-PAR			
	FHS-CONTROL-INFO			
	FHS-MAIN-RC	2	4.	primary return code
	FHS-ERROR-INFO			
	ERROR-CATEGORY	2	5.	secondary return code
	ERROR-REASON			
	PRINTER-RETURN-MSG			
	RETURN-MSG-TYPE	1	6.	type of acknowledgement: + or -
	RETURN-BYTE1	1	6.	printer return bytes
	RETURN-BYTE2	1	6.	
	RETURN-STATUS-INFO	2	6.	status byte
	FHS-OUTPUT-INFO			
	OUT-USER-AREA-LEN	4	7.	length of transfer-area
	OUT-USER-AREA-TRUNCATION	1	7.	LOW-VALUE or HIGH-VALUE
	FHS-INPUT-INFO			
	IN-PRINTER-RETURN-MSG	1	6.	HIGH-VALUE: printer-ack.
	IN-FIELD-DET	1	8.	LOW-VALUE or HIGH-VALUE
	IN-MSG-NILS	1	9.	LOW-VALUE or HIGH-VALUE
	IN-F-KEY	2	10.	n, if Fn key
	IN-K-KEY	2	11	n, if Kn key
	IN-USER-AREA-LEN	4	12	length of transfer area
	IN-MSG-LEN	4	13.	length of transferred data
	FHS-MAP-PAR			
	FHS-MAP-NAME	8	14.	„name-of-format“
	FHS-EXIT-MODE-NAME	8	15.	„name-of-exit-routine“
	...		16.	formatting specifications

FHS returns

TIAM CALL „WRTRD“

CALL	USING	Length	Descr.	Entry in data item
	FHS-MAP-LIB-OPT	1	17.	„Y“
	FHS-MAP-LIB-NAME	54	17.	„name-of-format-file“
	FHS-EXIT-LIB-OPT	1	15.	„Y“
	FHS-EXIT-LIB-NAME	54	15.	„file-for-exit-routine“
	FHS-EXIT-FOR-OUTPUT	1	15.	„Y“
	FHS-EXIT-FOR-INPUT	1	15.	„Y“
	FHS-MAP-OPTIONS	60	16.	formatting specifications
	...			

TIAM CALL „WRTRD“

7.3.2 DCAM calls

The format handling function is integrated into the DCAM COBOL calls

CALL YSEND

and

CALL YRECEIVE

These calls are described in the DCAM (TRANSDATA) manual “COBOL Calls”. Additional considerations regarding the use of FHS are given below.

Important

If the DCAM program is to provide for message I/O with the aid of the integrated Format Handling System (FHS), the following items must be supplied during connection setup (“YOPNCON”) or by means of the “Changing the Characteristics of a Connection” function (“YCHANGE”):

EDIT	with “**SYS**” for message editing by the system
EDITIN	with “**FOR**” for format handling during input
EDITOUT	with “**FOR**” for format handling during output

7.3.2.1 DCAM COBOL call for outputting formatted messages

```
CALL "YSEND" USING   APP-NAME
                     CONN-NAME
                     BEF-NAME
                     transfer-area
                     FHS-MAIN-PAR [(i)]
                     [additional-area].
```

where

APP-NAME, CONN-NAME and BEF-NAME are DCAM structures; they are described in the DCAM (TRANSDATA) manual “COBOL Calls”.

transfer-area

2 is the name of the output data transfer area.
In the case of CALL “YSEND”, the data transfer areas - as provided (with addressing aids) by IFG - must be preceded by an item in which FHS writes the length of the message. This can be done as follows:

```
01    transfer-area.
      03 length-field             PIC 9(5) COMP.
      03 format-name.
      COPY format-name.
```

[additional-area]

10 is the name of a terminal-specific administrative area which need only be specified when either partial formats or a restart is to be used. This administrative area must begin on a word boundary and, depending on the function desired, must have one of the following minimum lengths:

Partial formatting only	2028 bytes	for 2-Kbyte restart area
Restart only	2066 bytes	
Partial formatting and restart	4096 bytes	

Note

With formats having a large number of fields and/or with #formats, the minimum length for the restart area may be insufficient. In such a case, the size of the restart area must be increased appropriately.

The first two bytes of this area must contain the length of the area and the remainder must be deleted with LOW-VALUE.

The area is defined - e.g. for a restart - as follows:

```
01      additional-area.
    41  length-field           PIC 9(4) COMP SYNC  VALUE 2068.
    41  restart-area           PIC X(2066).
```

FHS-MAIN-PAR

controls the formatting. If the data structure is defined by means of tables and occurs several times, the appropriate index must also be specified.

Before the DCAM call CALL “**YSEND”** for outputting a formatted message is issued, the data transfer area and the items below must be supplied with information as follows:

1 The **FHS** item in the structure BEF-NAME **must** be given the value “**YES”** (for FHS application) (see the DCAM (TRANSDATA) manual “COBOL Calls”).

6 The **FHS-MAP-NAME** item in FHS-MAIN-PAR **must** contain the name of the format to be output.

In addition, the following optional entries in FHS-MAIN-PAR are possible:

9 If the formats are not stored in the standard format application file (F.MAPLIB), the following items must also be filled:

FHS-MAP-LIB-OPT with the value “**Y”** and **FHS-MAP-LIB-NAME** with the name of the format application file in which the desired format is stored as a module.

8 If the cursor is to be positioned explicitly and CALL “FHSCURS” has been called, the **FHS-MAP-CURSOR-OPT** item must be given the value “**Y”.**

8 If field attributes are to be modified in +formats (with CALL “FHSATTR” or a MOVE statement), the **FHS-MODY-ATTR** item must be given the value “**Y**” (does not apply to #formats).

8 If partial formats are to be used, the following items must also be filled:

- **FHS-PARTIAL-MAP-OPT** with the value “**Y”** and
- **FHS-MAP-PART** with the value “S” if it is not the last call in the partial formatting cycle or the value “L” if it is the last call in this cycle, and
- **MAP-CLEAR-OPT** with the value “Y” for the first call in a partial formatting cycle, and the value “N” for subsequent calls in this cycle (does not apply to #formats).

7 When an exit routine is used, the following items must also be filled:

- **FHS-EXIT-MOD-NAME** with the name of the exit routine,
- **FHS-EXIT-LIB-OPT** with a “Y” if the exit routine is not in the standard library F.EXITLIB,
- **FHS-EXIT-LIB-NAME** with the name of the module library containing the exit routine (only if FHS-EXIT-LIB-OPR=“Y”),
- **FHS-EXIT-FOR-OUTPUT** with a “Y” if the exit routine is to be called during output formatting,
- **FHS-EXIT-FOR-INPUT** with a “Y” if the exit routine is to be called during subsequent input formatting. (The user should bear in mind the next input entry as its FHS data structure will be the same as that used for the associated output.)

8 The remaining items of FHS-MAIN-PAR may be used to control further formatting characteristics (see page 303ff, FHS-MAIN-PAR).

FHS returns:

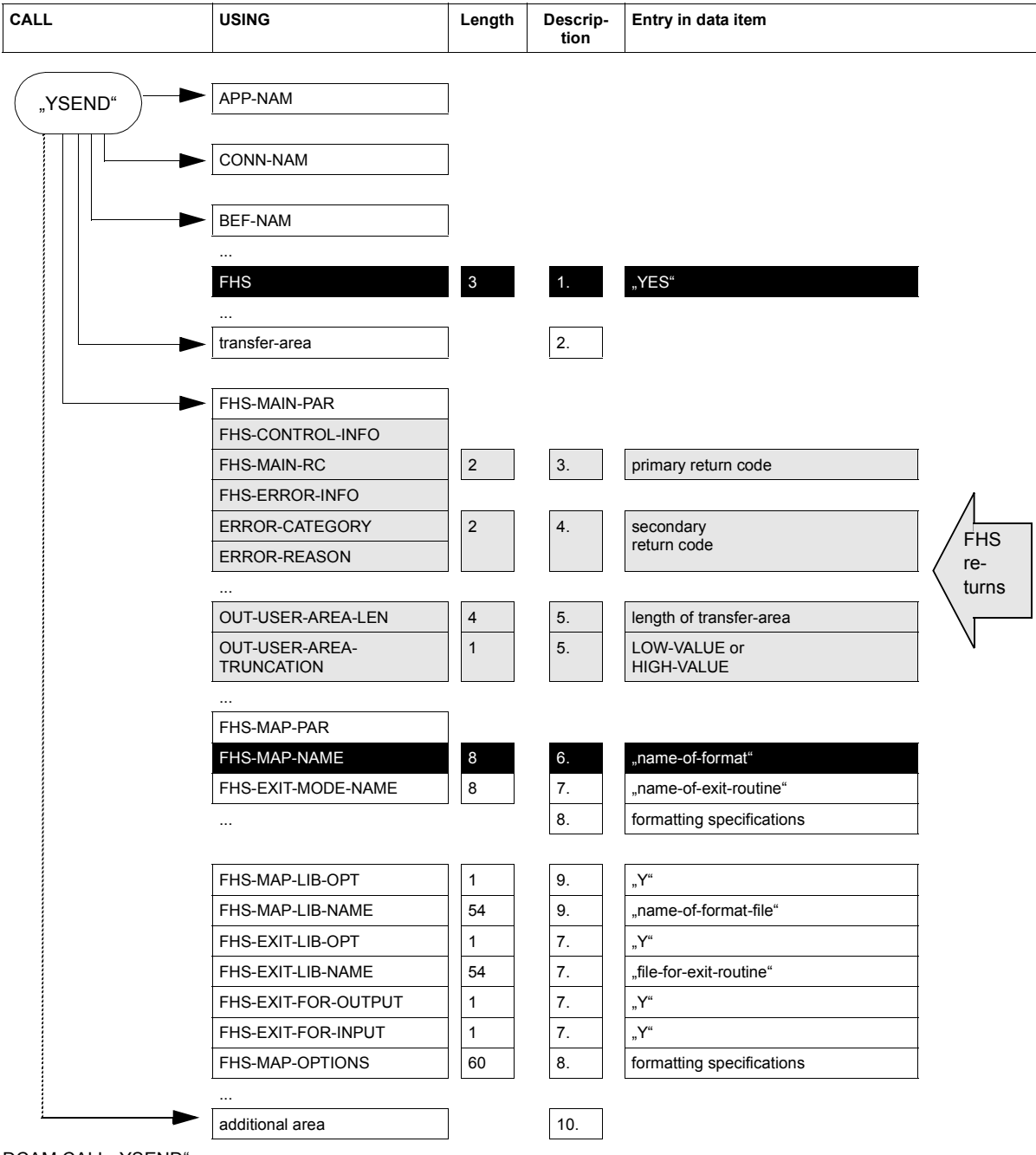
global and field attributes

3 the primary return code in **FHS-MAIN-RC** (see page 524ff),

4 the secondary return code in **ERROR-CATEGORY** and **ERROR-REASON** (see page 524ff),

5 the computed minimum length of the data transfer area in **OUT-USER-AREA-LEN** and whether the data transfer area is large enough in **OUT-USER-AREA-TRUNCATION**.

These items are evaluated by the user in addition to the DCAM return code following a YSEND call for formatted messages. The value 0 indicates that the formatting was error-free.



DCAM CALL „YSEND“

7.3.2.2 DCAM COBOL call for inputting formatted messages

```
CALL "YRECEIVE" USING   APP-NAME
                        CONN-NAME
                        BEF-NAME
                        transfer-area
                        FHS-MAIN-PAR
                        [additional-area].
```

where

APP-NAME, CONN-NAME and BEF-NAME are DCAM structures; they are described in the DCAM (TRANSDATA) manual “COBOL Calls”.

transfer-area

2 is the name of the input data transfer area.
In the case of CALL YRECEIVE, the data transfer areas - as provided (with addressing aids) by IFG - must be preceded by an item in which FHS writes the length of the message.
This can be done as follows:

```
01    transfer-area.
      03 length-field            PIC 9(4) COMP.
      03 format-name.
      COPY format-name.
```

FHS-MAIN-PAR

3 controls formatting.
It must be completely supplied with values prior to YSEND! For the relevant YSEND, DCAM internally stores this area and, upon CALL “YRECEIVE”, returns it to the specified area.

[additional-area]

14 is the name of a terminal-specific administrative area which need only be specified when either partial formats or a restart is to be used. This administrative area must be the same as that specified for the associated CALL “YSEND”.

Before the DCAM call CALL “**YRECEIVE”** for inputting formatted messages is issued, the following items must be filled:

1 The **FHS** item in the structure BEF-NAME **must** be given the value “**YES”** (for FHS application) (see the DCAM (TRANSDATA) manual “COBOL Calls”).

13 Since DCAM uses the same FHS data structure for this input as for the associated YSEND call, the entries in FHS-MAIN-PAR already exist. The FHS-EXIT-FOR-INPUT item (exit routine for input formatting) must be supplied when making the associated YSEND call.

FHS returns:

global and field attributes,

4 the primary return code in **FHS-MAIN-RC** (see page 524ff); if it has the value 0, the formatting operation was error-free.

5 the secondary return code in **ERROR-CATEGORY** and **ERROR-REASON** (see page 524ff).

6 whether the input message was an acknowledgment from the printer (**IN-PRINTER-RETURN-MSG**). If it is a printer acknowledgment, FHS also supplies the following items:

- **RETURN-MSG-TYPE** with the type of acknowledgment (+ve or -ve),
- **RETURN-BYTE1** and **RETURN-BYTE2** with the printer return bytes,
- **RETURN-STATUS-BYTE** with the contents of the status byte.

7 whether a field was detected during input formatting (**IN-FIELD-DET** item),

8 whether data was received (**IN-MSG-NILS** item),

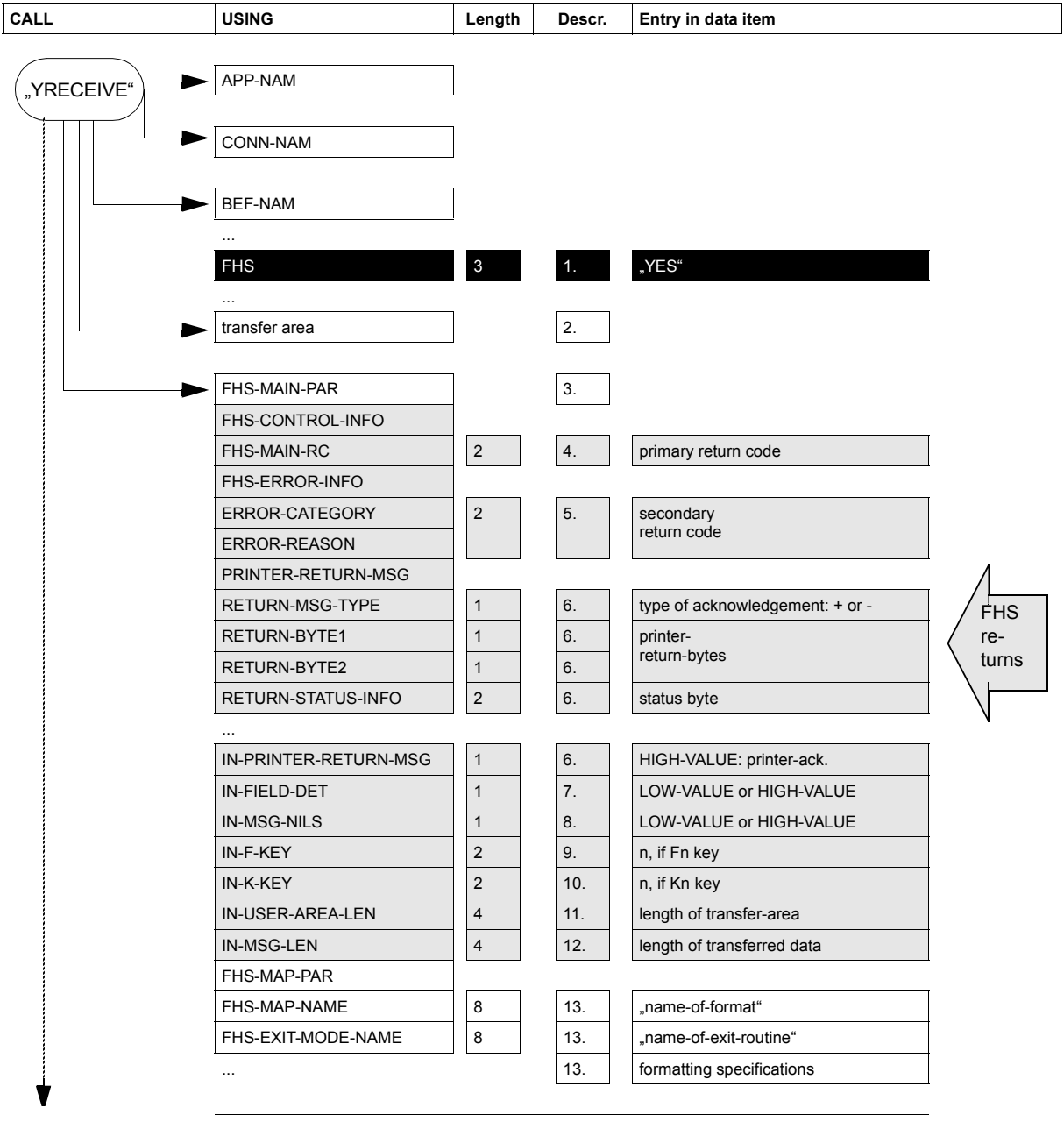
9 whether a function key was activated (**IN-F-KEY** item),

10 whether a short message was received (**IN-K-KEY** item),

11 the length of the data transfer area (**IN-USER-AREA-LEN** item),

12 the length of the input data (**IN-MSG-LEN** item).

These items are evaluated by the user in addition to the DCAM return code after a YRECEIVE call for formatted messages.



CALL	USING	Length	Descr.	Entry in data item
	FHS	3	1.	„YES“
	transfer area		2.	
	FHS-MAIN-PAR		3.	
	FHS-CONTROL-INFO			
	FHS-MAIN-RC	2	4.	primary return code
	FHS-ERROR-INFO			
	ERROR-CATEGORY	2	5.	secondary return code
	ERROR-REASON			
	PRINTER-RETURN-MSG			
	RETURN-MSG-TYPE	1	6.	type of acknowledgement: + or -
	RETURN-BYTE1	1	6.	printer-return-bytes
	RETURN-BYTE2	1	6.	
	RETURN-STATUS-INFO	2	6.	status byte
	IN-PRINTER-RETURN-MSG	1	6.	HIGH-VALUE: printer-ack.
	IN-FIELD-DET	1	7.	LOW-VALUE or HIGH-VALUE
	IN-MSG-NILS	1	8.	LOW-VALUE or HIGH-VALUE
	IN-F-KEY	2	9.	n, if Fn key
	IN-K-KEY	2	10.	n, if Kn key
	IN-USER-AREA-LEN	4	11.	length of transfer-area
	IN-MSG-LEN	4	12.	length of transferred data
	FHS-MAP-PAR			
	FHS-MAP-NAME	8	13.	„name-of-format“
	FHS-EXIT-MODE-NAME	8	13.	„name-of-exit-routine“
	...		13.	formatting specifications

FHS returns

DCAM CALL „YRECEIVE“

CALL	USING	Length	Descr.	Entry in data item
	FHS-MAP-LIB-OPT	1	13.	„Y“
	FHS-MAP-LIB-NAME	54	13.	„name-of-format-file“
	FHS-EXIT-LIB-OPT	1	13.	„Y“
	FHS-EXIT-LIB-NAME	54	13.	„file-for-exit-routine“
	FHS-EXIT-FOR-OUTPUT	1	13.	„Y“
	FHS-EXIT-FOR-INPUT	1	13.	„Y“
	FHS-MAP-OPTIONS	60	13.	formatting specification
	...			
→	additional area		14.	
	...			

DCAM CALL „YRECEIVE“

7.4 FHS COBOL calls

The following FHS COBOL calls are used to control formatting:

- **CALL “FHSCURS”** for explicit cursor positioning in +formats and *formats,
- **CALL “FHSATTR”** for field attribute updating in +formats and
- **CALL “FHSINIT”** for defining preset values for formatting.
- **CALL “FHSSERV”** for initializing the data transfer area with separate attribute blocks and field contents or for determining the character set name for 8-bit formats.

7.4.1 CALL “FHSCURS”

For formats that do not use the data transfer area with separate attribute blocks and field contents, it is possible by using CALL **“FHSCURS”** to position the cursor in any unprotected or detectable field of the format.

Format

```
CALL  "FHSCURS"   USING    FHS-MAIN-PAR [(i)]
                           field-name.
```

where

FHS-MAIN-PAR

contains the return code for the FHSCURS call. If the data structure is defined by means of tables and occurs several times, the appropriate index must be specified.

field-name

3 is the name of the field in which the cursor is to be positioned by FHS.

No items need to be written prior to CALL “FHSCURS”.

FHS returns

1 the primary return code in **FHS-MAIN-RC** (see page 524ff),

2 the secondary return code in **ERROR-CATEGORY** and **ERROR-REASON** (see page 524ff).

These items are evaluated by the user after an FHSCURS call. The value 0 indicates error-free execution of the FHSCURS call.

Note

If the cursor is to be positioned in a specific field during output, the FHSCURS subprogram must be called prior to output formatting. Then “Y” must be entered in the FHS-MAP-CURSOR-OPT item in FHS-MAIN-PAR. FHS will not position the cursor on the desired field unless these two conditions are satisfied.

Example

```
           .
           .
     CALL "FHSCURS"  USING FHS-MAIN-PAR
                           INPUT1.
     IF FHS-MAIN-RC NOT = 0  GO TO FHSERROR.
     MOVE "Y" TO FHS-MAP-CURSOR-OPT.
 OUTPUT1.
*
*      OUTPUT IN ACCORDANCE WITH ACCESS METHOD
*
           .
           .
```

CALL	USING	Length	Description	Entry in data item
“FHSCURS“	FHS-MAIN-PAR			
	FHS-CONTROL-INFO			
	FHS-MAIN-RC	2	1.	primary return code
	FHS-ERROR-INFO			
	ERROR-CATEGORY	2	2.	secondary return code
	ERROR-REASON			
	...			
	item-name		3.	

FHS returns

FHS COBOL CALL “FHSCURS”

7.4.2 CALL “FHSATTR”

For formats that do not use the data transfer area with separate attribute blocks and field contents, it is possible by using the FHSATTR subprogram to modify the attribute fields of a field in the data transfer area and thus to update the attributes of this field. For #formats, this subprogram is of significance only when using the field attribute group ‘Attribute Combination’. This function is also performed by the “FHS-ATTRIBUT-MOVE” copy elements; if they exhibit a suitable attribute combination, the FHSATTR call can be replaced by a simple MOVE statement.

Format

```
CALL  "FHSATTR"  USING  FHS-CONTROL-INFO [(i)]
                        FHS-ATTR-PAR [(i)]
                        attribute-field.
```

where

FHS-CONTROL-INFO

is that part of the FHS-MAIN-PAR data structure which contains the return codes. If the data structure is defined by means of tables and occurs several times, the appropriate index must be specified.

FHS-ATTR-PAR

controls attribute updating. If the data structure is defined by means of tables and occurs several times, the appropriate index must be specified.

attribute-field

7 is the name of the attribute field to be updated.

Before the CALL “FHSATTR” for attribute updating is issued, the following items of the data structure FHS-ATTR-PAR must be written for the desired attributes:

4 The **A-PROT-LEVEL** item with any of the values “UNPR” (unprotected), “PROT” (protected), “PRET” (protected, returned upon input), “FSET” (unprotected, returned upon input) or “PDET” (detectable),

5 the **A-DISP-LEVEL** item with any of the values “B” (bright), “H” (normal) or “D” (invisible).

6 (optional:) the items **A-NO-HARDCOPY** (non-printing), **A-NUMERIC** (numeric), **A-SIGNAL** (flashing), **A-ITALIC** (italics), **A-WIDE** (wide type) or **A-TALL** (tall type) with the value “Y”.

3 The item A-UPDATE-METHOD must have the value “REP” (default); any other value causes the FHSATTR call to terminate, i.e. the attribute field will not be updated.

FHS returns:

1 the primary return code in **FHS-MAIN-RC** (see page 524ff).

2 the secondary return code in **ERROR-CATEGORY** and **ERROR-REASON** (see page 524ff).

These items are evaluated by the user after an FHSATTR call. The value 0 indicates error-free execution of the FHSATTR call.

CALL	USING	Length	Descrip-tion	Entry in data item
"FHSATTR"	FHS-CONTROL-INFO			
	FHS-MAIN-RC	2	1.	primary return code
	FHS-ERROR-INFO			
	ERROR-CATEGORY	2	2.	secondary return code
	ERROR-REASON			
	...			
	FHS-ATTR-PAR			
	A-UPDATE-METHODE	3	3..	"REP"
	A-PROT-LEVEL	4	4..	"UNPR" / "PROT" / "PRET" /FSET" / "PDET"
	A-DISP-LEVEL	1	5..	"B"/"H"/"D"
	A-NO-HARDCOPY	1	6..	"Y"
	A-NUMERIC	1	6..	"Y"
	A-SIGNAL	1	6..	"Y"
	A-ITALIC	1	6..	"Y"
	A-WIDE	1	6..	"Y"
	A-TALL	1	6..	"Y"
	...			
	attribute-field		7.	

FHS returns

FHS COBOL CALL "FHSATTR"

Note

- In an FHSATTR call, all attributes of a field must be specified; non-specified attributes are assigned default values. It is not sufficient to specify only those attributes that are to be updated.
- The attributes for output formatting and those for input formatting must be compatible.
- All attribute fields that are not to be updated must be set to LOW-VALUE; otherwise FHS will not use the field attributes from the format definition. Note that if the same data transfer area is used for both input and output formatting, FHS overwrites the attribute fields with the length of transmitted data.
- If FHS is to use the field attributes from the attribute fields in the data transfer area rather than from the format definition, the **FHS-MODY-ATTRS** item in FHS-MAIN-PAR must be set to “Y” prior to any output formatting that will involve field attribute updating. If, for further formatting operations, FHS is to use the field attributes from the format description, the FHS-MODY-ATTRS must be set to “N”.
- CALL “FHSATTR” can be replaced by a simple MOVE statement if the copy element FHS-ATTRIBUT-MOVE contains an appropriate attribute combination. For attribute modification by a MOVE statement see the section starting on page 381.

Example

An invalid input is to be returned underlined to the display terminal.

```
           .
           .
     MOVE LOW-VALUE TO OUTPUTA.
     MOVE "UNPR" TO A-PROT-LEVEL
     MOVE "B"    TO A-DISP-LEVEL
     MOVE "Y"    TO A-ITAL.
     CALL "FHSATTR"  USING  FHS-CONTROL-INFO
                            FHS-ATTR-PAR
                            INPUTA.
     IF FHS-MAIN-RC NOT = 0  GO TO FHSERROR.
     MOVE "Y" TO FHS-MODY-ATTRS.
  OUTPUT1.
*
*      OUTPUT IN ACCORDANCE WITH THE ACCESS METHOD
*
     IF FHS-MAIN-RC NOT = 0  GO TO FHSERROR.
     MOVE "N" TO FHS-MODY-ATTRS.
           .
           .
```

7.4.3 CALL “FHSINIT”

The FHSINIT subprogram serves to initialize formatting and to specify which formats are to be loaded on opening formatting. For #formats, the start parameters are issued here. CALL “FHSINIT” is necessary

- if you want to work with formats that are to be loaded on opening or
- more than 100 different formats are to be used.

Furthermore, the FHSINIT call also enables the user to define his own formatting standard. This standard can be modified for every formatting operation.

If CALL “FHSINIT” is repeated several times, the data area FHS-INIT-PAR-GENERAL is not evaluated for calls after the first call since the formatting operation was already initiated with the first CALL “FHSINIT”.

Format

```
CALL  "FHSINIT"  USING  FHS-CONTROL-INFO [(i)]
                        FHS-INIT-PAR [(i)]
                        area
                        [format-list].
```

where

FHS-CONTROL-INFO

is the part of the FHS-MAIN-PAR data structure containing the return codes. If the data structure is defined by means of tables and occurs several times, the appropriate index must be specified.

FHS-INIT-PAR

defines the formatting standard. If the data structure is defined by means of tables and occurs several times, the appropriate index must be specified.

area

7 is the name of any area which must be specified for compatibility reasons. Even though this area is not used it must be specified for each FHSINIT call.

[format-list]

8 is the name of a list containing the names of the formats that are to be loaded on opening formatting. Each entry in this list must be 8 bytes long. The number of entries is defined in FHS-INIT-PAR in the item FHS-RES-MAP-NO. If the latter has the value 0, FHS does not evaluate this list.

Before the CALL "**FHSINIT"** for formatting initialization is issued, the following items must be filled:

3 **FHS-RES-MAP-NO** with the number of formats if you wish to load formats on opening formatting. If FHS-RES-MAP-NO ≠ 0, FHS expects, as the third operand of USING, the name of a list containing the names of the formats to be loaded.

4 **FHS-MAP-NO** with the maximum number of formats used when the capacity of the directory for format entries is other than 100 entries.

6 If the formats are not located in the standard format application file (F.MAPLIB), the following items must also be filled:

FHS-MAP-LIB-OPT with "**Y**" and **FHS-MAP-LIB-NAME** with the name of the format application file in which the formats are stored as modules.

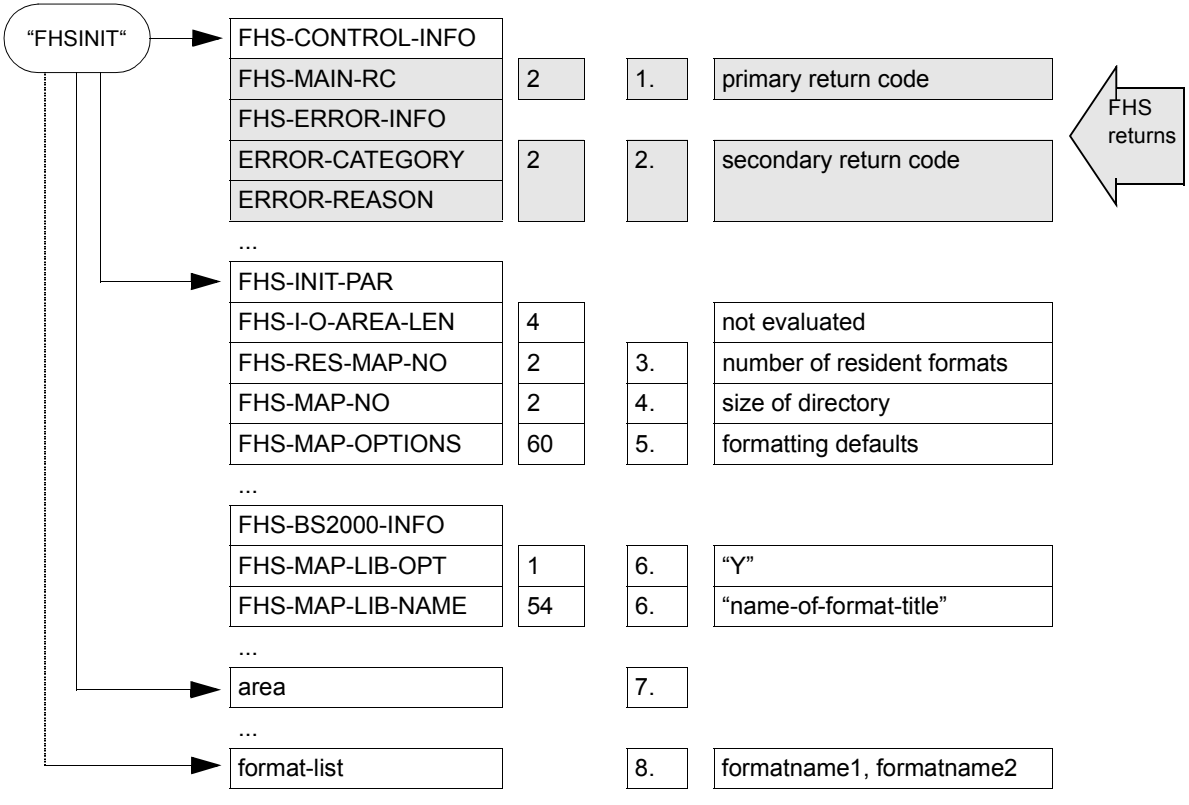
5 The remaining items of FHS-INIT-PAR can be utilized to define your own formatting standard (see page 330ff, FHS-INIT-PAR). For every formatting operation, this standard can be updated in the FHS-MAIN-PAR.

FHS returns:

1 the primary return code in **FHS-MAIN-RC** (see page 524ff)

2 the secondary return code in **ERROR-CATEGORY** and **ERROR-REASON** (see page 524ff)

These items are evaluated by the user after an FHSINIT call. The value 0 indicates error-free execution of the FHSINIT call.



CALL	USING	Length	Description	Entry in data item
	FHS-CONTROL-INFO			
	FHS-MAIN-RC	2	1.	primary return code
	FHS-ERROR-INFO			
	ERROR-CATEGORY	2	2.	secondary return code
	ERROR-REASON			
	...			
	FHS-INIT-PAR			
	FHS-I-O-AREA-LEN	4		not evaluated
	FHS-RES-MAP-NO	2	3.	number of resident formats
	FHS-MAP-NO	2	4.	size of directory
	FHS-MAP-OPTIONS	60	5.	formatting defaults
	...			
	FHS-BS2000-INFO			
	FHS-MAP-LIB-OPT	1	6.	"Y"
	FHS-MAP-LIB-NAME	54	6.	"name-of-format-title"
	...			
	area		7.	
	...			
	format-list		8.	formatname1, formatname2

FHS COBOL CALL "FHSINIT"

Example

```
          .
          .
WORKING-STORAGE SECTION.
01  INOUT               PIC X.
01  FORMATLIST
    02  FORMATNAME      PIC X(8). OCCURS 3 TIMES.
*
        COPY FHSMAINP.
        COPY FHSINITP.
             .
             .
PROCEDURE DIVISION.
    MOVE "3" TO FHS-RES-MAP-NO
    MOVE "FORMAT1" TO FORMATNAME (1)
    MOVE "FORMAT2" TO FORMATNAME (2)
    MOVE "FORMAT3" TO FORMATNAME (3).
    MOVE "Y"    TO FHS-MAP-LIB-OPT
    MOVE "$PTS.F.MAPLIB" TO FHS-MAP-LIB-NAME.
    CALL "FHSINIT"  USING  FHS-CONTROL-INFO
                           FHS-INIT-PAR
                           AREA
                           FORMATLIST.
    IF FHS-MAIN-RC NOT = 0  GO TO FHSERROR.
          .
          .
```

7.4.4 CALL “FHSSERV”

The FHSSERV subprogram allows you to utilize four FHS service functions:

– ’Initialization of the Data Transfer Area’ for #formats,

– ’Determine name of character set’

– ‘Unload Format’

– ‘Dynamic Retrieval of Information on the Structure of the Addressing Aid for #Formats’

7.4.4.1 Initialization of the Data Transfer Area

Here all field attributes are filled in accordance with their default values in the format. Neither the global attributes (apart from ’Formatting acknowledgment’) nor the field contents are updated. It is thus possible to reset to the initial status data transfer areas that have already been supplied with data.

Format

```
CALL  "FHSSERV"   USING   FHS-MAIN-PAR [(i)]
                            transfer-area
```

where:

FHS-MAIN-PAR

contains the data structure FHS-MAP-PAR for specifying function parameters and FHS-CONTROL-INFO for the return codes.

transfer-area

6 is the name of the transfer area without the preceding length field. This is determined by the addressing aid, as provided by IFG. The preceding length field is dependent on the access method. With CALL “FHSSERV” the access method is not known.

Before CALL “FHSSERV” the following items must be filled:

4 item **FHS-SERVICE-FUNCTION** with **1** (number of the service function),

3 item **FHS-MAP-NAME** with the name of the format for which the service function is to be executed.

In addition, the following optional entries can be made in FHS-MAIN-PAR:

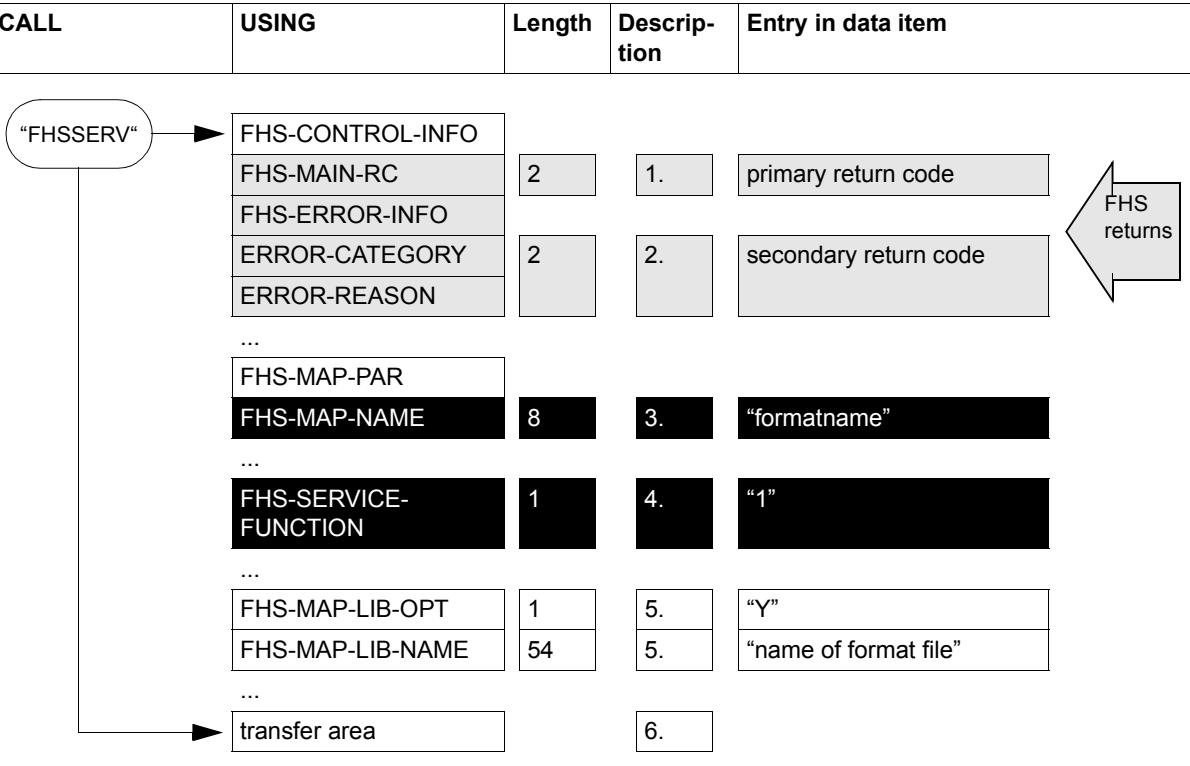
5 If the formats are not located in the standard format application file (F.MAPLIB), the following items must also be filled:

FHS-MAP-LIB-OPT with "**Y**" and **FHS-MAP-LIB-NAME** with the name of the format application file in which the desired format is stored as a module.

FHS returns:

1 the primary return code in **FHS-MAIN-RC** (see page 524ff)

2 the secondary return code in **ERROR-CATEGORY** and **ERROR-REASON** (see page 524ff)



FHS COBOL CALL "FHSSERV" for initialization

7.4.4.2 Determine name of character set

The format specified in FHS-MAIN-PAR is loaded and the name of the corresponding character set is entered in the FHS-CCSN-INFO field of the data structure FHS-CCSN-PAR.

Format

```
CALL  "FHSSERV"   USING   FHS-MAIN-PAR [(i)]
                          FHS-CCSN-PAR [(i)]
```

where

FHS-MAIN-PAR

contains the data structure FHS-MAP-PAR for specifying the function parameters and FHS-CONTROL-INFO for the return codes.

FHS-CCSN-PAR

6 contains the parameter list for the character set names.

Before the CALL “FHSSERV” is issued, the following items must be filled:

4 **FHS-SERVICE-FUNCTION** with **2** (number of the service function),

3 **FHS-MAP-NAME** with the name of the format for which the service function is to be executed.

You can also enter the following in FHS-MAIN-PAR:

5 If the formats are not located in the standard format application (F.MAPLIB), the following items must also be filled:

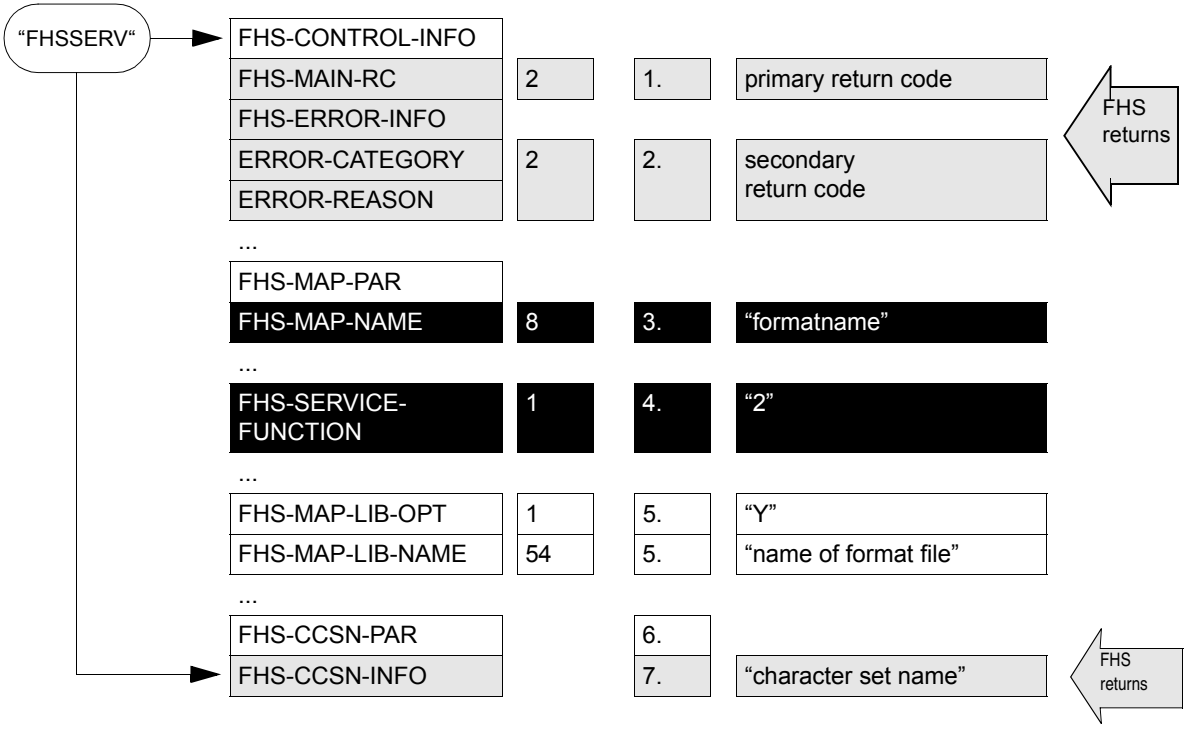
FHS-MAP-LIB-OPT with “**Y”** and **FHS-MAP-LIB-NAME** with the name of the format application file in which the desired format is stored as a module.

FHS returns:

1 the primary return code in **FHS-MAIN-RC** (see page 524ff).

2 the secondary return code in **ERROR-CATEGORY** and **ERROR-REASON** (see page 524ff).

7 the name of the character set in **FHS-CCSN-INFO** of the data structure FHS-CCSN-PAR. Blanks are entered for a 7-bit format.



FHS COBOL CALL "FHSSERV" for determining the name of the character set.

Example

```
      .
      .
WORKING-STORAGE SECTION.
01 fhs working storage section.
   COPY FHSMAINP.
01 character set area.
   COPY FHSCCSNP.
      .
      .
PROCEDURE DIVISION.
   MOVE "Y"           TO FHS-MAP-LIB-OPT          IN FHSMAINP.
   MOVE library-name  TO FHS-MAP-LIB-NAME         IN FHSMAINP.
   MOVE 2             TO FHS-SERVICE-FUNCTION     IN FHSMAINP.
   MOVE format-name   TO FHS-MAP-NAME             IN FHSMAINP.
   CALL "FHSSERV" USING FHS-MAIN-PAR  FHS-CCSN-PAR.
   DISPLAY "NAME OF CHARACTER SET: " FHS-CCSN-INFO IN FHSCCSNP UPON TERMINAL.
      .
      .
```

7.4.4.3 Unload format

The format specified in FHS-MAIN-PAR is unloaded and can be replaced by a modified format.

Format

```
CALL  "FHSSERV"   USING   FHS-MAIN-PAR [(i)]
                          transfer-area
```

where

FHS-MAIN-PAR

contains the data structure FHS-MAP-PAR for specifying the function parameters and FHS-CONTROL-INFO for the return codes.

transfer-area

is a dummy field and is not used.

Before the CALL “FHSSERV” is issued, the following items must be filled:

4 **FHS-SERVICE-FUNCTION** with **5** (number of the service function),

3 **FHS-MAP-NAME** with the name of the format for which the service function is to be executed.

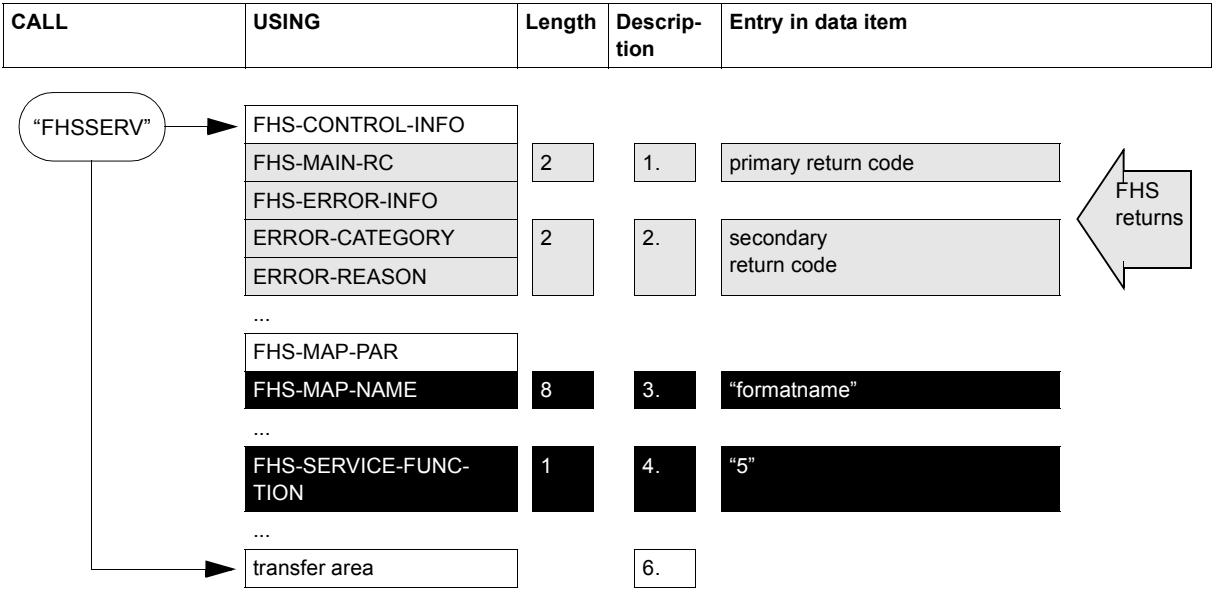


Diagram FHS-COBOL-CALL “FHSSERV” for unloading formats

7.4.4.4 Dynamically retrieving information on the structure of the addressing aid for #formats

The structure of the addressing aid for the format specified in FHS-MAIN-PAR is returned in the transfer area.

Format

```
CALL  "FHSSERV"   USING   FHS-MAIN-PAR [(i)]
                          transfer-area.
```

where

FHS-MAIN-PAR

contains the data structure FHS-MAP-PAR for specifying the function parameters and FHS-CONTROL-INFO for the return codes.

transfer-area

must be supplied with a description of the information to be provided by FHS as well as an indication of where the function result is to be stored (see section “Dynamically retrieving information on the structure of the addressing aid for #formats” on page 255).

Before the CALL “FHSSERV” is issued, the following items must be filled:

4 **FHS-SERVICE-FUNCTION** with **3** (number of the service function)

3 **FHS-MAP-NAME** with the name of the format for which the service function is to be executed.

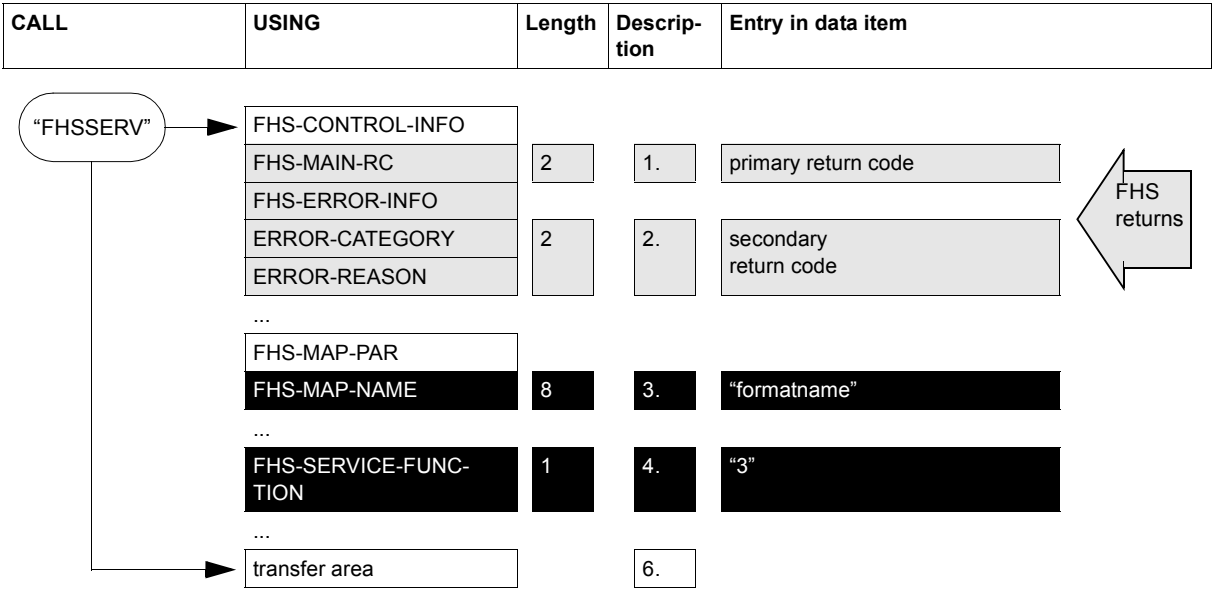


Diagram FHS-COBOL-CALL “FHSSERV” for dynamic retrieval of information on the structure of the addressing aid for #formats

7.5 Attribute updating

7.5.1 Attribute value list FHS-ATTRIBUTE-VALUES

FHS-ATTRIBUTE-VALUES is the following attribute value list, which is copied into the application program with **COPY "FHSAVAL".** It generates symbolically addressable attribute values for the global attributes and field attributes of the data transfer area with separate attribute blocks and field contents.

```
*****************************************************************
* NAME              FHSAVAL                                     *
* VERSION           811                                         *
*                                                               *
*    Attribute values for IFG addressing aids                   *
*                                                               *
* END-INTERFACE     FHSAVAL                                     *
*****************************************************************
 01  FHS-ATTRIBUTE-VALUES.
*****************************************************************
*
*   GLOBAL ATTRIBUTE VALUES (CHARACTERS)
*   ------------------------------------
*    GA-DEFAULT-VALUES.
           02  GA-DEFAULT              PIC X VALUE SPACE.
*    FORMATTING-INDICATORS.
*       FIELDS-MODIFICATION.
           02  GA-MODIFIED             PIC X VALUE „Y“.
           02  GA-NOT-MODIFIED         PIC X VALUE „ „.
*       FIELDS-DETECTION.
           02  GA-DETECTED             PIC X VALUE „Y“.
           02  GA-NOT-DETECTED         PIC X VALUE „ „.
*       FIELDS-VALIDATION.
           02  GA-VALID                PIC X VALUE „V“.
           02  GA-NOT-VALID            PIC X VALUE „ „.
*       FIELDS-UNDEFINED.
           02  GA-UNDEFINED            PIC X VALUE „Y“.
           02  GA-NOT-UNDEFINED        PIC X VALUE „ „.
*    INPUT-IDENTIFICATION.
*       INPUT-KEY-CLASS.
           02  GA-INPUT-KEY            PIC X VALUE „I“.
           02  GA-F-KEY                PIC X VALUE „F“.
           02  GA-K-KEY                PIC X VALUE „K“.
           02  GA-POS-RM               PIC X VALUE „P“.
           02  GA-NEG-RM               PIC X VALUE „N“.
           02  GA-INPUT-NONE           PIC X VALUE „ „.
```

```
*    DEVICE-CONTROLS.
*       INIT-CONTROL.
           02  GA-NO-INIT             PIC X VALUE „N“.
           02  GA-FIRST-INIT          PIC X VALUE „F“.
           02  GA-LAST-INIT           PIC X VALUE „L“.
           02  GA-BOTH-INIT           PIC X VALUE „B“.
*       TABULATOR-CONTROL.
           02  GA-AUTO-TAB            PIC X VALUE „A“.
           02  GA-NO-AUTO-TAB         PIC X VALUE „N“.
*       FUNCTION-LOCK.
           02  GA-KEYLOCK             PIC X VALUE „K“.
*       VMI-CONTROL.
           02  GA-VMI-1               PIC X VALUE „1“.
           02  GA-VMI-2               PIC X VALUE „2“.
           02  GA-VMI-3               PIC X VALUE „3“.
*       HMI-CONTROL.
           02  GA-HMI-1               PIC X VALUE „1“.
           02  GA-HMI-2               PIC X VALUE „2“.
           02  GA-HMI-3               PIC X VALUE „3“.
*    OUTPUT-CONTROLS.
*       CYCLE-CONTROL.
           02  GA-CLOSE               PIC X VALUE „C“.
*       COPY-CONTROL.
           02  GA-HARDCOPY-GEN        PIC X VALUE „H“.
           02  GA-HARDCOPY-LOC        PIC X VALUE „L“.
*       ALARM-CONTROL.
           02  GA-ALARM               PIC X VALUE „A“.
*       HOLE-COLOR.
           02  GA-NO-COLOR            PIC X VALUE „U“.
           02  GA-GREY-HOLE           PIC X VALUE „G“.
           02  GA-WHITE-HOLE          PIC X VALUE „W“.
*    FORMATTING-CONTROLS.
*       DISPLAY-SELECTION.
           02  GA-BOXB                PIC X VALUE „B“.
           02  GA-BOXL                PIC X VALUE „C“.
           02  GA-KEB                 PIC X VALUE „K“.
           02  GA-KEL                 PIC X VALUE „L“.
*       LEVEL-SELECTION.
           02  GA-LEVEL-1             PIC X VALUE „1“.
           02  GA-LEVEL-2             PIC X VALUE „2“.
           02  GA-LEVEL-3             PIC X VALUE „3“.
           02  GA-LEVEL-P             PIC X VALUE „P“.
*       OUTPUT-MODE.
           02  GA-RDIF                PIC X VALUE „R“.
*       CURSOR-CONTROL.
           02  GA-FIELD-CURSOR        PIC X VALUE „F“.
           02  GA-EDIT-CURSOR         PIC X VALUE „E“.
           02  GA-REL-CURSOR          PIC X VALUE „R“.
```

```
*       USER-EXIT-CONTROL.
           02  GA-NO-UEXIT            PIC X VALUE „N“.
           02  GA-OUT-UEXIT           PIC X VALUE „O“.
           02  GA-IN-UEXIT            PIC X VALUE „I“.
           02  GA-BOTH-UEXIT          PIC X VALUE „B“.
/
*   FIELD ATTRIBUTE VALUES (CHARACTERS)
*   -----------------------------------
*    DEFAULT-VALUES.
           02  FA-DEFAULT             PIC X VALUE SPACE.
*
*    BASIC-ATTRIBUTES.
*       INPUT-STATE-/-INPUT-STATE-ACT.
           02  FA-MODIFIED            PIC X VALUE „M“.
           02  FA-CLEARED             PIC X VALUE „C“.
           02  FA-DETECTED            PIC X VALUE „D“.
           02  FA-UNDEFINED           PIC X VALUE „U“.
           02  FA-NOT-TOUCHED         PIC X VALUE „ „.
*       EDIT-STATE.
           02  FA-VALID               PIC X VALUE „V“.
           02  FA-INVALID             PIC X VALUE „I“.
           02  FA-MUST-ERROR          PIC X VALUE „M“.
           02  FA-NOT-CHECKED         PIC X VALUE „ „.
*       OUTPUT-CONTROL.
           02  FA-OUTPUT-INIT         PIC X VALUE „I“.
           02  FA-OUTPUT-DATA         PIC X VALUE „D“.
           02  FA-OUTPUT-UNDEFINED    PIC X VALUE „U“.
*
*    FIELD-INPUT.
*       INPUT-CONTROL.
           02  FA-NORMAL-IN           PIC X VALUE „N“.
           02  FA-MUST-IN             PIC X VALUE „M“.
           02  FA-POTMUST-IN          PIC X VALUE „P“.
           02  FA-AUTORET-IN          PIC X VALUE „A“.
*       PROTECTION.
           02  FA-UNPROTECTED         PIC X VALUE „U“.
           02  FA-PROTECTED           PIC X VALUE „P“.
           02  FA-ASKIP               PIC X VALUE „A“.
           02  FA-DETECTABLE          PIC X VALUE „D“.
*
*    DISPLAY-CONTROL.
*       INTENSITY.
           02  FA-HIGH-INTENSITY      PIC X VALUE „H“.
           02  FA-NORMAL-INTENSITY    PIC X VALUE „N“.
*       VISIBILITY.
           02  FA-VISIBLE             PIC X VALUE „V“.
           02  FA-SIGNALING           PIC X VALUE „S“.
           02  FA-INVISIBLE           PIC X VALUE „I“.
```

```
*       UNDERLINE.
           02  FA-UNDERLINED          PIC X VALUE „Y“.
           02  FA-NOT-UNDERLINED      PIC X VALUE „N“.
*       INVERSE.
           02  FA-INVERSE             PIC X VALUE „Y“.
           02  FA-NOT-INVERSE         PIC X VALUE „N“.
*
*    COLOUR.
           02  FA-RED                 PIC X VALUE „1“.
           02  FA-GREEN               PIC X VALUE „2“.
           02  FA-YELLOW              PIC X VALUE „3“.
           02  FA-BLUE                PIC X VALUE „4“.
           02  FA-MAGENTA             PIC X VALUE „5“.
           02  FA-CYAN                PIC X VALUE „6“.
           02  FA-WHITE               PIC X VALUE „7“.
           02  FA-NO-COLOUR           PIC X VALUE „N“.
*
*    CURSOR.
           02  FA-CURSOR              PIC X VALUE „Y“.
           02  FA-HOLD-CURSOR         PIC X VALUE „H“.
           02  FA-NO-CURSOR           PIC X VALUE „N“.
*****************************************************************
```

See page 46ff for a description of the data items.

7.5.2 Copy element FHS-ATTRIBUTE-MOVE

This section describes the updating of attributes for formats that do not use the data transfer area with separate attribute blocks and field contents. For #formats, this section is of significance only when using the field attribute group ‘Attribute Combination’ (see page 62ff).

FHS offers the COBOL programmer two options for updating the attributes of the format fields in +formats in his program:

- CALL “FHSATTR” (see page 363ff) or
- the **FHS-ATTRIBUTE-MOVE** copy elements; if a suitable attribute combination exists in this data structure, it can be transferred to the desired attribute field by means of a simple MOVE statement.

FHS-ATTRIBUTE-MOVE is the following data structure, which is copied into the application program by means of **COPY “FHSATTRM”.**

```
01  ATTR-MOVE-AREA.
    COPY FHSATTRM.
*****************************************************************
*                                                               *
* NAME              FHSATTRM                                    *
* VERSION           811                                         *
*                                                               *
*    Data structure to move attributes                          *
*                                                               *
* END-INTERFACE     FHSATTRM                                    *
*****************************************************************
** KCALPH   ** UNPROT,BRT,PRINT
               41 KYCALPH      PIC 9(5) COMP VALUE 20512.
               41 KXCALPH      REDEFINES KYCALPH.
                  42 FILLER    PIC 99.
                  42 KCALPH    PIC 9(4) COMP.
** KCNUME   ** UNPROT,BRT,NUM
               41 KYCNUME      PIC 9(5) COMP VALUE 21024.
               41 KXCNUME      REDEFINES KYCNUME.
                  42 FILLER    PIC 99.
                  42 KCNUME    PIC 9(4) COMP.
** KCPROT   ** PROT,NORM
               41 KCPROT       PIC 9(4) COMP VALUE  4360.
** KCUNPR   ** UNPROT,BRT
               41 KYCUNPR      PIC 9(5) COMP VALUE 20512.
               41 KXCUNPR      REDEFINES KYCUNPR.
                  42 FILLER    PIC 99.
                  42 KCUNPR    PIC 9(4) COMP.
** KCNINT   ** UNPROT,NORM
               41 KYCNINT      PIC 9(5) COMP VALUE 20488.
               41 KXCNINT      REDEFINES KYCNINT.
                  42 FILLER    PIC 99.
                  42 KCNINT    PIC 9(4) COMP.
** KCDINT   ** UNPROT,DRK
               41 KYCDINT      PIC 9(5) COMP VALUE 20484.
               41 KXCDINT      REDEFINES KYCDINT.
                  42 FILLER    PIC 99.
                  42 KCDINT    PIC 9(4) COMP.
** KCHINT   ** UNPROT,BRT
               41 KYCHINT      PIC 9(5) COMP VALUE 20512.
               41 KXCHINT      REDEFINES KYCHINT.
                  42 FILLER    PIC 99.
                  42 KCHINT    PIC 9(4) COMP.
** KCITAL   ** UNPROT,BRT,ITAL
               41 KYCITAL      PIC 9(5) COMP VALUE 20514.
               41 KXCITAL      REDEFINES KYCITAL.
                  42 FILLER    PIC 99.
                  42 KCITAL    PIC 9(4) COMP.
```

```
** KCSIGN   ** UNPROT,BRT,SIGN
               41 KYCSIGN      PIC 9(5) COMP VALUE 20513.
               41 KXCSIGN      REDEFINES KYCSIGN.
                  42 FILLER    PIC 99.
                  42 KCSIGN    PIC 9(4) COMP.
** KCDETE   ** PROT,BRT,DET
               41 KCDETE       PIC 9(4) COMP VALUE  6432.
** KCPREM   ** FSET,BRT
               41  KCPREM      PIC 9(4) COMP VALUE  5152.
** KCAUN    ** UNPROT,NORM
               41 KYCAUN       PIC 9(5) COMP VALUE 20488.
               41 KXCAUN       REDEFINES KYCAUN.
                  42 FILLER    PIC 99.
                  42 KCAUN     PIC 9(4) COMP.
** KCNUN    ** UNPROT,NORM,NUM
               41 KYCNUN       PIC 9(5) COMP VALUE 21000.
               41 KXCNUN       REDEFINES KYCNUN.
                  42 FILLER    PIC 99.
                  42 KCNUN     PIC 9(4) COMP.
** KCAPN    ** PROT,NORM
               41 KCAPN        PIC 9(4) COMP VALUE  4360.
** KCNPN    ** PROT,NORM,NUM
               41 KCNPN        PIC 9(4) COMP VALUE  4872.
** KCAUD    ** UNPROT,DRK
               41 KYCAUD       PIC 9(5) COMP VALUE 20484.
               41 KXCAUD       REDEFINES KYCAUD.
                  42 FILLER    PIC 99.
                  42 KCAUD     PIC 9(4) COMP.
** KCNUD    ** UNPROT,DRK,NUM
               41 KYCNUD       PIC 9(5) COMP VALUE 20996.
               41 KXCNUD       REDEFINES KYCNUD.
                  42 FILLER    PIC 99.
                  42 KCNUD     PIC 9(4) COMP.
** KCAPD    ** PROT,DRK
               41 KCAPD        PIC 9(4) COMP VALUE  4356.
** KCNPD    ** PROT,DRK,NUM
               41 KCNPD        PIC 9(4) COMP VALUE  4868.
** KCAUH    ** UNPROT,BRT
               41 KYCAUH       PIC 9(5) COMP VALUE 20512.
               41 KXCAUH       REDEFINES KYCAUH.
                  42 FILLER    PIC 99.
                  42 KCAUH     PIC 9(4) COMP.
** KCNUH    ** UNPROT,BRT,NUM
               41 KYCNUH       PIC 9(5) COMP VALUE 21024.
               41 KXCNUH       REDEFINES KYCNUH.
                  42 FILLER    PIC 99.
                  42 KCNUH     PIC 9(4) COMP.
```

```
** KCAPH    ** PROT,BRT
              41 KCAPH        PIC 9(4) COMP VALUE  4384.
** KCNPH    ** PROT,BRT,NUM
              41 KCNPH        PIC 9(4) COMP VALUE  4896.
** KCAUI    ** UNPROT,BRT,ITAL
              41 KYCAUI       PIC 9(5) COMP VALUE 20514.
              41 KXCAUI       REDEFINES KYCAUI.
                 42 FILLER    PIC 99.
                 42 KCAUI     PIC 9(4) COMP.
** KCNUI    ** UNPROT,BRT,ITAL,NUM
              41 KYCNUI       PIC 9(5) COMP VALUE 21026.
              41 KXCNUI       REDEFINES KYCNUI.
                 42 FILLER    PIC 99.
                 42 KCNUI     PIC 9(4) COMP.
** KCAPI    ** PROT,NORM,ITAL
              41 KCAPI        PIC 9(4) COMP VALUE  4362.
** KCNPI    ** PROT,NORM,ITAL,NUM
              41  KCNPI       PIC 9(4) COMP VALUE  4874.
** KCAUS    ** UNPROT,BRT,SIGN
              41 KYCAUS       PIC 9(5) COMP VALUE 20513.
              41 KXCAUS       REDEFINES KYCAUS.
                 42 FILLER    PIC 99.
                 42 KCAUS     PIC 9(4) COMP.
** KCNUS    ** UNPROT,BRT,SIGN,NUM
              41 KYCNUS       PIC 9(5) COMP VALUE 21025.
              41 KXCNUS       REDEFINES KYCNUS.
                 42 FILLER    PIC 99.
                 42 KCNUS     PIC 9(4) COMP.
** KCAPS    ** PROT,NORM,SIGN
              41 KCAPS        PIC 9(4) COMP VALUE  4361.
** KCNPS    ** PROT,NORM,SIGN,NUM
              41 KCNPS        PIC 9(4) COMP VALUE  4873.
** KCAUND   ** UNPROT,NORM,DET
              41 KYCAUND      PIC 9(5) COMP VALUE 22536.
              41 KXCAUND      REDEFINES KYCAUND.
                 42 FILLER    PIC 99.
                 42 KCAUND    PIC 9(4) COMP.
** KCNUND   ** UNPROT,NORM,DET
              41 KYCNUND      PIC 9(5) COMP VALUE 22536.
              41 KXCNUND      REDEFINES KYCNUND.
                 42 FILLER    PIC 99.
                 42 KCNUND    PIC 9(4) COMP.
** KCAPND   ** PROT,NORM,DET
              41 KCAPND       PIC 9(4) COMP VALUE  6408.
** KCNPND   ** PROT,NORM,DET,NUM
              41 KCNPND       PIC 9(4) COMP VALUE  6920.
** KCAUHD   ** UNPROT,BRT,DET
              41 KYCAUHD      PIC 9(5) COMP VALUE 22560.
```

```
                  41 KXCAUHD      REDEFINES KYCAUHD.
                     42 FILLER    PIC 99.
                     42 KCAUHD    PIC 9(4) COMP.
** KCNUHD   ** UNPROT,BRT,DET
                  41 KYCNUHD      PIC 9(5) COMP VALUE 22560.
                  41 KXCNUHD      REDEFINES KYCNUHD.
                     42 FILLER    PIC 99.
                     42 KCNUHD    PIC 9(4) COMP.
** KCAPHD   ** PROT,BRT,DET
                  41 KCAPHD       PIC 9(4) COMP VALUE  6432.
** KCNPHD   ** PROT,BRT,DET,NUM
                  41 KCNPHD       PIC 9(4) COMP VALUE  6944.
** KCAUID   ** UNPROT,BRT,DET,ITAL
                  41 KYCAUID      PIC 9(5) COMP VALUE 22562.
                  41 KXCAUID      REDEFINES KYCAUID.
                     42 FILLER    PIC 99.
                     42 KCAUID    PIC 9(4) COMP.
** KCNUID   ** UNPROT,BRT,DET,ITAL
                  41 KYCNUID      PIC 9(5) COMP VALUE 22562.
                  41 KXCNUID      REDEFINES KYCNUID.
                     42 FILLER    PIC 99.
                     42 KCNUID    PIC 9(4) COMP.
** KCAPID   ** PROT,NORM,DET,ITAL
                  41 KCAPID       PIC 9(4) COMP VALUE  6410.
** KCNPID   ** PROT,NORM,DET,ITAL,NUM
                  41 KCNPID       PIC 9(4) COMP VALUE  6922.

** KCAUSD   ** UNPROT,BRT,DET
                  41 KYCAUSD      PIC 9(5) COMP VALUE 22560.
                  41 KXCAUSD      REDEFINES KYCAUSD.
                     42 FILLER    PIC 99.
                     42 KCAUSD    PIC 9(4) COMP.
** KCNUSD   ** UNPROT,BRT,DET
                  41 KYCNUSD      PIC 9(5) COMP VALUE 22560.
                  41 KXCNUSD      REDEFINES KYCNUSD.
                     42 FILLER    PIC 99.
                     42 KCNUSD    PIC 9(4) COMP.
** KCAPSD   ** PROT,NORM,DET
                  41 KCAPSD       PIC 9(4) COMP VALUE  6408.
** KCNPSD   ** PROT,NORM,DET,NUM
                  41 KCNPSD       PIC 9(4) COMP VALUE  6920.
** KCAUNP   ** FSET,NORM
                  41 KCAUNP       PIC 9(4) COMP VALUE  5128.
** KCNUNP   ** FSET,NORM,NUM
                  41 KCNUNP       PIC 9(4) COMP VALUE  5640.
```

```
** KCAPNP   ** PROTRET,NORM
              41 KYCAPNP      PIC 9(5) COMP VALUE 12296.
              41 KXCAPNP      REDEFINES KYCAPNP.
                 42 FILLER    PIC 99.
                 42 KCAPNP    PIC 9(4) COMP.
** KCNPNP   ** PROTRET,NORM,NUM
              41 KYCNPNP      PIC 9(5) COMP VALUE 12808.
              41 KXCNPNP      REDEFINES KYCNPNP.
                 42 FILLER    PIC 99.
                 42 KCNPNP    PIC 9(4) COMP.
** KCAUHP   ** FSET,BRT
              41 KCAUHP       PIC 9(4) COMP VALUE  5152.
** KCNUHP   ** FSET,BRT,NUM
              41 KCNUHP       PIC 9(4) COMP VALUE  5664.
** KCAPHP   ** PROTRET,BRT
              41 KYCAPHP      PIC 9(5) COMP VALUE 12320.
              41 KXCAPHP      REDEFINES KYCAPHP.
                 42 FILLER    PIC 99.
                 42 KCAPHP    PIC 9(4) COMP.
** KCNPHP   ** PROTRET,BRT,NUM
              41 KYCNPHP      PIC 9(5) COMP VALUE 12832.
              41 KXCNPHP      REDEFINES KYCNPHP.
                 42 FILLER    PIC 99.
                 42 KCNPHP    PIC 9(4) COMP.
** KCPBSP   ** PROT,BRT,SIGN
              41 KCPBSP       PIC 9(4) COMP VALUE  4385.
************************************************************
```

Which copy element (KCxxxx) you must specify for which attributes is marked in the table with gray shading.

The following table shows for which combinations of attributes in this data structure there are corresponding elements and gives the names of the elements:

Fieldname	Field attributes														
	PROT	UNPROT	SIGN	DET	NUM	PRINT	BRT	NORM	DRK	IC	ITAL	WIDE	TALL	FSET	PROTRET
	protected	unprotected	flashing	detectable	numeric	printable	bright	normal	invisible	cursor	italics/ underlined	wide, spaced	tall	unprotected	protected
KCALPH		X				X	X								
KCNUME		X			X	X	X								
KCPROT	X					X		X							
KCUNPR		X				X	X								
KCNINT		X				X		X							
KCDINT		X				X			X						
KCHINT		X				X	X								
KCITAL		X				X	X				X				
KCSIGN		X	X			X	X								
KCDETE	X			X		X	X								
KCAUN		X				X		X							
KCNUN		X			X	X		X							
KCAPN	X					X		X							
KCNPN	X				X	X		X							
KCAUD		X				X			X						
KCNUD		X			X	X			X						
KCAPD	X					X			X						
KCNPD	X				X	X			X						
KCAUH		X				X	X								
KCNUH		X			X	X	X								
KCAPH	X					X	X								
KCNPH	X				X	X	X								
KCAUI		X				X	X				X				
KCNUI		X			X	X	X				X				
KCAPI	X					X		X			X				

Fieldname	Field attributes														
	PROT	UNPROT	SIGN	DET	NUM	PRINT	BRT	NORM	DRK	IC	ITAL	WIDE	TALL	FSET	PROTRET
	protected	unprotected	flashing	detectable	numeric	printable	bright	normal	invisible	cursor	italics/ underlined	wide, spaced	tall	unprotected	protected
KCNPI	X				X	X		X			X				
KCAUS		X	X			X	X								
KCNUS		X	X		X	X	X								
KCAPS	X		X			X		X							
KCNPS	X		X		X	X		X							
KCPREM						X	X							X	
KCAUNP						X		X						X	
KCNUNP					X	X		X						X	
KCAPNP						X		X							X
KCNPNP					X	X		X							X
KCAUHP						X	X							X	
KCNUHP					X	X	X							X	
KCAPHP						X	X								X
KCNPHP					X	X	X								X
KCAUND		X		X		X		X							
KCNUND		X		X		X		X							
KCAPND	X			X		X		X							
KCNPND	X			X	X	X		X							
KCAUHD		X		X		X	X								
KCNUHD		X		X		X	X								
KCAPHD	X			X		X	X								
KCNPHD	X			X	X	X	X								
KCAUID		X		X		X	X				X				
KCNUID		X		X		X	X				X				
KCAPID	X			X		X		X			X				
KCNPID	X			X	X	X		X			X				

Fieldname	Field attributes														
	PROT	**UNPROT**	**SIGN**	**DET**	**NUM**	**PRINT**	**BRT**	**NORM**	**DRK**	**IC**	**ITAL**	**WIDE**	**TALL**	**FSET**	**PROTRET**
	protected	**unprotected**	**flashing**	**detectable**	**numeric**	**printable**	**bright**	**normal**	**invisible**	**cursor**	**italics/ underlined**	**wide, spaced**	**tall**	**unprotected**	**protected**
KCAUSD		X		X		X	X								
KCNUSD		X		X		X	X								
KCAPSD	X			X		X		X							
KCNPSD	X			X	X	X		X							

Note on the 3270

A combination of PROT and NUM or PROTRET and NUM produces the ASKIP function.

Note

- Insure that the attributes for output formatting and those for input formatting are compatible.
- All attribute fields that are not to be updated must be set to LOW-VALUE; otherwise FHS will not use the field attributes from the format description. Note that if the same data transfer area is used for both input and output formatting, FHS overwrites the attribute fields with the length of the transferred data (depending on MAP-EFF-LEN=Y).
- If FHS is to use the field attributes from the attribute fields in the data transfer area rather than from the format definition, the **FHS-MODY-ATTRS** item in FHS-MAIN-PAR must be set to “Y” prior to any output formatting that will involve field attribute updating. If, for further formatting operations, FHS is to use the field attributes from the format description, then FHS-MODY-ATTRS must be set to “N”.

Example

The example on page 363 could look like this when the MOVE statement is used:

```
           .
           .
     MOVE LOW-VALUE TO EINGABEA.
     MOVE KCSIGN TO OUTPUTA.
*             UNPROT, BRT AND SIGN
     MOVE "Y" TO FHS-MODY-ATTRS.
  OUTPUT1.
*
*       OUTPUT IN ACCORDANCE WITH ACCESS METHOD
*
           .
```

7.6 Using exit routines in COBOL programs

Exit routines serve to check format fields for certain contents and to update these contents. The format definition using IFG defines which fields in the exit routine are to be evaluated. Moreover, for each field an exit remark is defined which is transferred to the exit routine by FHS.

Structure of a COBOL exit routine

- **DATA DIVISION**

  In the **LINKAGE SECTION**, the data structure FHS-EXITMOD-PAR is copied to the exit routine by means of the statement

```
01     exit-area.
       COPY FHSEXITP.
```

- **PROCEDURE DIVISION**

  This must begin with

```
PROCEDURE DIVISION  USING  exit-area.
```

  For input formatting, an optional item defined in the LINKAGE SECTION may be specified under USING. This item contains the address of the length field in the case of input formatting.
  In the exit routine, a return code to be evaluated by the user program can be stored in EXIT-RET-INFO.
  Return from the exit routine is effected by the statement

```
EXIT PROGRAM.
```

  On return from the exit routine, FHS transfers the contents of the EXIT-DATA item to the data field for which the exit routine was invoked; the length of transfer is equivalent to the length of the data field.

The same exit routine can be used for both input and output. The number of operands specified under USING is not checked.

If you wish to evaluate the contents of the data item (EXIT-DATA) in a COBOL exit routine, note that FHS enters the data in EXIT-DATA only as far as the data field extends; the remainder of EXIT-DATA stays the same (it is not blank-filled). Therefore, EXIT-DATA can be evaluated only up to data field length. This can be achieved, for example, by redefining the EXIT-DATA item in the LINKAGE SECTION by means of the REDEFINES clause for every data field to be evaluated (see also the following example of an exit routine).

Note

When you use exit routines and a restart area, the data updates performed in the exit routine are not taken into account in the restart area in the case of output formatting. For #formats, see page 76.

Example of an exit routine

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  FHSEXIT.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 LINKAGE SECTION.
 01  EXITAREA.
     COPY FHSEXITP.
     42   FIELD1   REDEFINES   EXIT-DATA.
          43       INPUT1      PIC X(4).
          43       FILLER      PIC X(76).
     42   FIELD2   REDEFINES   EXIT-DATA.
          43       SUFFIX      PIC X(30).
          43       FILLER      PIC X(50).
     42   FIELD3   REDEFINES   EXIT-DATA.
          43       OUTPUT1     PIC X(55).
          43       FILLER      PIC X(25).
*
 PROCEDURE DIVISION USING EXITAREA.
     IF EXIT-IN-OUT = "O" GO TO OUTPUT1.
*    FIELDTYPE = IN1 ?
     IF EXIT-IDENT = "IN1" GO TO IN1.
*    FIELDTYPE = IN2 ?
     IF EXIT-IDENT = "IN2" GO TO IN2.
     GO TO EXIT1.
 IN1.
     IF INPUT1 = "????"  MOVE "1" TO EXIT-RET-INFO.
     GO TO EXIT1.
 IN2.
     IF SUFFIX = "********" MOVE "2" TO EXIT-RET-INFO.
     GO TO EXIT1.
 OUTPUT1.
     IF EXIT-IDENT NOT = "OUT" GO TO EXIT1.
     IF OUTPUT1 = "@@@@@@@@"  MOVE "OUTPUT1" TO EXIT-DATA.
 EXIT1.
     EXIT PROGRAM.
```

How do you use an exit routine?

If an exit routine is to be used, the following items of FHS-MAIN-PAR must be filled prior to the I/O call (with integrated formatting):

- **FHS-EXIT-MOD-NAME** with the name of the exit routine
- **FHS-EXIT-FOR-OUTPUT** with “Y” if the exit routine is to be activated for output formatting (ignored with #formats),
- **FHS-EXIT-FOR-INPUT** with “Y” if the exit routine is to be activated for input formatting (ignored with #formats),
- **FHS-EXIT-LIB-OPT** with “Y” if the exit routine is not contained in the standard library F.EXITLIB,
- **FHS-EXIT-LIB-NAME** with the name of the module library containing the exit routine.

The exit routine will then be called during formatting after the editing of the field by the formatting routine for each field for which this was specified in the format definition.

After formatting, the **EXIT-RET-INFO** item contains the return code set in the exit routine. If the exit routine is called several times, EXIT-RET-INFO contains the return code last set.

See page 76 and the global attributes ‘User Exit Control’ and ‘User Exitroutine RC’ for how to proceed in the case of #formats.

Example

```
           .
           .
    MOVE "FHSEXIT"     TO FHS-EXIT-MOD-NAME
    MOVE "Y"           TO FHS-EXIT-LIB-OPT
    MOVE "FHS.MODLIB" TO FHS-EXIT-LIB-NAME
    MOVE "Y"           TO FHS-EXIT-FOR-OUTPUT
    MOVE "Y"           TO FHS-EXIT-FOR-INPUT.
 OUTPUT1.
*
*      OUTPUT IN ACCORDANCE WITH THE ACCESS METHOD
*
    IF EXIT-RET-INFO = "1" THEN  . . .
*      EVALUATE RETURN CODE OF EXIT ROUTINE
           .
           .
```

When using an exit routine, note that:

– Unlike in ASSEMBLER, FHS COBOL exit routines are **not** linked to the application program but must be present as a module in a module library (normally F.EXITLIB). At run time they are loaded from this library.

– If the exit routine is also written in COBOL, it too needs modules from the COBOL runtime system. These modules must be linked together with the exit routine and placed in the module library. The following procedure links the exit routine FHSEXIT, which resides in the module library FHS.MODLIB, with the COBOL runtime modules into one module and stores this module in FHS.MODLIB.

```
/BEGIN-PROC
/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/DELETE-SYSTEM-FILE FILE-NAME=OMF
/START-PROGRAM $TSOSLNK
MOD FHSEXIT
RESOLVE ,$RZ.COBOL85.LZS
INCLUDE FHSEXIT,FHS.MODLIB
END
/MODIFY-JOB-SWITCHES ON=1
/START-PROGRAM $LMS
LIB FHS.MODLIB,OUT
PAR OVERWRITE=YES
ADDR *OMF
END
/ASSIGN-SYSDTA TO-FILE=*PRIMARY
/MODIFY-JOB-SWITCHES OFF=1
/END-PROC
```

Unicode formats are only supported with COBOL2000 compiler.

7.7 Using partial formats

Page 36 describes what partial formats are and gives an example of how they can be used.

Partial formatting for output

A number of partial formats can be sent to the terminal at a time. All the partial formats being output at a given same time are formatted in a “partial formatting cycle”. A partial formatting cycle consists of a number of calls for formatted output (CALL “YSEND” for DCAM and CALL “WROUT” or CALL “WRTRD” for TIAM) for the partial formats to be output together.

The **FHS-MAP-PART** field in FHS-MAIN-PAR must be supplied with one of the values “**S**” or “**L**”:

“S” The output call is valid for one partial format within the partial formatting cycle; it is not the last call in this cycle. The partial format is only formatted. It is not output.

“L” The following output call is the last or only one in this partial formatting cycle. The partial format is formatted and output together with the other partial formats in the cycle.

CAUTION!

In partial formatting not every output call (CALL “YSEND”, CALL “WROUT” or CALL “WRTRD”) initiates output. Only in the last output call of a partial formatting cycle, if FHS-MAP-PART has the value “L”, are all the partial formats sent to the terminal. In the other output calls of the output cycle (where FHS-MAP-PART=“S”) FHS only formats the partial format and TIAM or DCAM does **not** output the message. FHS controls this with FHS-MAIN-RC 24. The access methods then issue a return code, that refers to the FHS return code, to the application program. This must be evaluated.

TIAM example

The partial formats TF1, TF2, TF3 and TF4 are to be sent to the terminal in a partial formatting cycle.

```
      .
      .
      .
 PROCEDURE DIVISION.
      MOVE "F" TO EDIT-MODE IN EDIT-OUT
      MOVE "TF1" TO FHS-MAP-NAME
      MOVE "Y" TO FHS-PARTIAL-MAP-OPT
      MOVE "S" TO FHS-MAP-PART
      MOVE "Y" TO MAP-CLEAR-OPTION
*     F O R M A T T I N G      T F 1
      CALL "WRTRD" USING TIAM-CONTROL-INFO
                         FORMAT1-UA
                         FORMAT1-4-UA
                         FHS-MAIN-PAR.
      MOVE "N" TO MAP-CLEAR-OPTION
      MOVE "TF2" TO FHS-MAP-NAME
*     F O R M A T T I N G      T F 2
      CALL "WRTRD" USING TIAM-CONTROL-INFO
                         FORMAT2-UA
                         FORMAT1-4-UA
                         FHS-MAIN-PAR.
      MOVE "TF3" TO FHS-MAP-NAME.
*     F O R M A T T I N G      T F 3
      CALL "WRTRD" USING TIAM-CONTROL-INFO
                         FORMAT3-UA
                         FORMAT1-4-UA
                         FHS-MAIN-PAR.
      MOVE "L" TO FHS-MAP-PART
      MOVE "TF4" TO FHS-MAP-NAME.
*     F O R M A T T I N G    T F 4   AND OUTPUT ALL PARTIAL FORMATS
      CALL "WRTRD" USING TIAM-CONTROL-INFO
                         FORMAT4-UA
                         FORMAT1-4-UA
                         FHS-MAIN-PAR.
       .
       .
       .
```

Not until the last CALL “WRTRD” is the format output and read in again after input on the terminal. With this input FHS formats the first partial format for which data is available.

Partial formatting for input

With the first call for the formatted input of partial formats (CALL “YRECEIVE” for DCAM or CALL “WRTRD” for TIAM) the access method reads in the message and FHS formats the first partial format for which data is available. The following applies to #formats:

– For TIAM:
  The partial #format into which data was entered at the data display terminal is read.

– For openUTM:
  All partial #formats are read, including those which do not expect input and those into which nothing has been entered.

The other partial formats can then be formatted with additional calls as follows:

CALL “YRECEIVE” for DCAM and

CALL “WRTRD” for TIAM, provided TIAM is informed via the field FHS-MAP-PART (FHS-MAP-PART=“N”).

Only the **first** input call actually initiates input; all further calls merely effect a serial input formatting of the partial formats. After formatting the name of the formatted partial format is entered in the field FHS-MAP-NAME. TIAM and DCAM perform no input or output. FHS controls this with FHS-MAIN-RC 24. The access methods then issue a return code, that refers to the FHS return code, to the application program. This must be evaluated.

Note

For the input formatting of partial formats a common data transfer area should be used and this then transferred in accordance with the format name supplied to the format-specific data transfer area.

TIAM example

```
           .
           .
           .
           MOVE "Y" TO FHS-PARTIAL-MAP-OPT.
           .     ⎫
           .     ⎬ partial formatting cycle for output
           .     ⎭
*    LAST WRTRD CALL IN THIS PARTIAL FORMATTING CYCLE          *
           MOVE "L" TO FHS-MAP-PART.
           CALL "WRTRD" USING TIAM-CONTROL-INFO
                              FORMAT4-UA
                              FORMAT1-4-UA
                              FHS-MAIN-PAR.
           IF FHS-MAP-NAME=SPACES GOTO FINISH
      *                                                          *
      *    PROCESSING OF PARTIAL FORMAT WHOSE NAME IS IN         *
      *    FHS-MAP-NAME.                                         *
      *                                                          *
           MOVE "N" TO FHS-MAP-PART
           CALL "WRTRD" USING TIAM-CONTROL-INFO
                              FORMAT4-UA
                              FORMAT1-4-UA
                              FHS-MAIN-PAR.
           .
           .
           .
          IF FHS-NAME=SPACES GOTO FINISH.
      *                                                          *
      *   PROCESSING OF PARTIAL FORMAT WHOSE NAME IS IN          *
      *   FHS-MAP-NAME.                                          *
      *                                                          *
           .
           .
           .
          FINISH
```

Notes on the use of partial formats

– The start line number is ignored when partial formats are output on a printer.

– FHS-MAPPING-METHOD=“ONLY” can always be used when outputting partial formats, even if the partial format has not been output before. FHS then replaces the value internally by FHS-MAPPING-METHOD=“BEGN”.

– If FHS-PARTIAL-MAP-OPT does not have the value “Y” and the format was not generated as a partial format (definition of a start line number), normal formatting is used, taking account of the start line number. FHS supplies the following return code:

  ```
  FHS-MAIN-RC=8, FHS-ERROR-CATEGORY=36, FHS-ERROR-REASON=104.
  ```

– If the format is not a partial format and FHS-PARTIAL-MAP-OPT has the value “Y”, the format is formatted normally.

– For partial formatting, in DCAM COBOL programs the application program must define a terminal-specific administrative area (MAPLIST area). This area must be specified under USING in CALL “YSEND” and CALL “YRECEIVE”. This area must begin on a word boundary and contain its length in the first two bytes; the remainder must be deleted with “LOW-VALUE” before it is used for the first time. Thereafter this area must not be accessed by the application program. The area is required by FHS for the management of partial formats and as a restart area for the restart function. The following minimum lengths apply:

for partial formatting without a restart:	2028 bytes
for partial formatting with a restart: (using a restart area of 2 Kbytes)	4096 bytes

  With formats having a large number of fields and/or with #formats, the minimum length for the restart area may be insufficient. In such a case, the size of the restart area must be increased appropriately.

  FHS checks for partial formatting whether the administrative area was specified and if its length is sufficient. In the event of an error, FHS sends a return code.
  You define this administrative area as follows:

  ```
  01      additional-area.
      41  length-field    PIC 9(4) COMP SYNC VALUE 2068.
      41  restart-area    PIC X(2066).
  ```

– For the use of partial formats that use the data transfer area with separate attribute blocks and field contents, see also page 78.

– Note that not all data in FHS-MAIN-PAR can be modified in one partial formatting cycle. Refer to the following table for the restrictions involved:

Item in FHS-MAIN-PAR	Modifying permitted?	Remarks
FHS-MAPPING-METHOD	yes	Every partial format can be output with FHS-MAPPING-METHOD=“ONLY” (instead of “BEGN”). FHS-MAPPING-METHOD=“RSET” is only possible if output formatting has already been performed for the format.
MAP-SCREEN-PRE-MOD	no	Functions as specified in the first partial formatting call of the output cycle. Any change in the subsequent partial formatting calls is ignored; a return code is output (warning).
MAP-POS-DET-CHAR MAP-NEG-DET-CHAR	yes	
MAP-BEL-OPTION	yes	An audible alarm is triggered for each partial formatting call with MAP-BEL-OPTION=“Y”.
FHS-MODY-ATTRS	yes	FHS records the values in the MAPLIST area.
MAP-USE-ALL-ATTRS	yes	
MAP-READ- METHOD	no	Rejected with a return code.
MAP-READ- NILS	no	Rejected with a return code.
FHS-EXIT-FOR-INPUT FHS-EXIT-FOR-OUTPUT	yes	
MAP-HARDCOPY-OPTION	no	Functions as specified in the first partial formatting call of the output cycle; ignored for subsequent partial calls.
MAP-AUTO-HARDCOPY	no	Functions as specified in the first partial formatting call of the output cycle; ignored for subsequent partial calls.
MAP-CLEAR-OPTION	yes	For the first partial formatting call of the output cycle MAP-CLEAR-OPTION must have the value “Y”; for subsequent calls it must have the value “N”.
MAP-LOCK-KEYS	no	Functions as specified in the first partial formatting call of the output cycle; ignored for subsequent partial formatting calls.
MAP-AUTO-TAB	no	Functions as specified in the first partial formatting call of the output cycle; ignored for subsequent partial formatting calls.
MAP-RESTART-OPT1	no	Rejected with a return code.
MAP-PARTIAL-OPT	no	The value of FHS-PARTIAL-MAP-OPT must be “Y”.
FHS-MAP-PART	yes	For the last partial formatting run of the output cycle, FHS-MAP-PART must have the value “L”; for all other runs it must be “S”. Exception: sequential input formatting using TIAM with FHS-MAP-PART=“N”.
MAP-DEVICE-CLASS	no	Rejected with a return code.
MAP-EFF-LEN	yes	

Item in FHS-MAIN-PAR	Modifying permitted?	Remarks
MAP-LIB-LOAD-MODE MAP-LIB-LOAD-FILE	--	Decentralized format handling not possible with partial formatting.
MAP-PRINT-LINES MAP-PRINT-COLUMNS MAP-PRINT-PAPER MAP-PRINT-FORM	--	Partial formatting not possible for output on a printer.
MAP-PRINTER-OPTION	no	Functions as specified in the first partial formatting call of the output cycle; ignored for subsequent partial formatting calls.

7.8 Compiling and linking FHS COBOL programs

Compiling FHS COBOL programs

Before the compiler is called, you need to assign the library from which the COBOL compiler will take the copy elements required in the program. The following must be present in this library:

– the FHS data structures required for your program,

– the data structures required by the access method used,

– the addressing aids.

Example

```
/DELETE-SYSTEM-FILE FILE-NAME=OMF
/ASSIGN-SYSDTA TO-FILE=filename
/SET-FILE-LINK LINK-NAME=COBLIB,FILE-NAME=COBLIB
/START-PROGRAM $RZ.COBOL85
```

Refer to the BS2000 COBOL manuals for how to proceed if the copy elements are contained in different libraries.

Linking FHS COBOL programs

The following modules must be linked with your application program:

- the formatting module **MFHSCALL** and the FHS CALLs
- with **TIAM**, the module **DCCOBRTS** for formatted input/output
- with **DCAM**, the modules of the DCAM COBOL interface.

All TIAM modules are supplied in the library SYSLIB.TIAM, and all DCAM modules in the library SYSLIB.DCAM.

In addition, the modules from the COBOL runtime system are needed, which are linked by means of the RESOLVE statements.

The exit routine is not linked but stored in a module library.

Example

With the following procedure you can link the FHS application program FHSPROG, which is available as a module in the module library FHS.MODLIB.

```
/BEGIN-PROC
/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/START-PROGRAM $TSOSLNK
PROG FHSPROG,FILENAM=FHSPROG.LOAD
INCLUDE FHSPROG,FHS.MODLIB
INCLUDE MFHSCALL,FHS.MFHSCALL
RESOLVE ,$TSOS.SYSLIB.DCAM
RESOLVE ,$RZ.COBOL85.LZS
END
/ASSIGN-SYSDTA TO-FILE=*PRIMARY
/END-PROC
```

7.9 Addressing aids in COBOL

If the Procedure Division of the program is written in COBOL (e.g. to utilize FHS COBOL calls), the addressing aids must also be written in COBOL. This is a standard function in IFG.

When copying the addressing aids to the application program, the user must insure that the addressing aids are preceded by a 4-byte (TIAM) or 2-byte (DCAM) length field. The COPY statement in the WORKING-STORAGE SECTION will then have the following format:

```
01    user-area.
      03 length field             PIC 9(n) COMP.
      COPY  [x]formatname.
```

or, if different transfer areas are used for input and output:

```
01    formatnameI.
      03 length-field-i           PIC 9(n) COMP.
      COPY  [x]formatnameI.
01    formatnameO.
      03 length-field-o.          PIC 9(n) COMP.
      COPY  [x]formatnameO.
```

where

n is 5 for TIAM,
is 4 for DCAM,

x (optional) is the prefix specified in IFG.

7.10 Sample program with FHS COBOL

The following sample program shows how, in the case of several fields featuring edit checking (validation), a correction can be requested by the terminal operator; various methods are used for the different fields:

– field unchanged, cursor in first errored field, message,

– field underlined, cursor in first errored field, message,

– field unchanged and cursor in first errored field.

The third course of action is that automatically taken by FHS if the format is output unchanged as an update output (see page 77ff).

The sample program refers to the following format:

```
                              O R D E R   E N T R Y

Customer no.: _____________                          Data:____________
Order no.: ___________

_____________________________________________________________________________

  Item Designation                                  Qty   Price      Total
  ___  ___________________________________________  ###   ________   @@@@@@@@@@@@@@

     Enter 'end' to terminate program: ________

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.     PRGCOB2B.
 DATE-WRITTEN.   07-12-92.
*
*
*  DATA TRANSFER AREA                    (#FORMAT)
*       WITH SEPARATE ATTRIBUTE BLOCKS AND FIELD CONTENTS
*
*
*  ATTRIBUTE MODIFICATIONS
*       VIA GLOBAL ATTRIBUTES OR FIELD ATTRIBUTES
*
*
*  SAMPLE PROGRAM FOR TIAM
*       OUTPUT/INPUT OF A FORMAT, UNTIL ALL FIELDS ARE VALID,
*       THEN CALCULATE TOTAL.
*       ORDER = Y  ---> NEW SCREEN
*       ORDER = N  +  FUNCTION = END  --->  PROGRAM END
*
*
*  SPECIAL CHARACTERISTICS OF THE INDIVIDUAL FIELDS:
*
*    CUSTNO    = MANDATORY INPUT
*    DATE      = CURRENT DATE WITH CALENDAR CHECK
*    ORDNO     = MANDATORY INPUT
*    ITEM      = MANDATORY INPUT
*    DESIG     = MANDATORY INPUT,
*                MINIMUM INPUT LENGTH
*    QTY       = ARITHMETIC, ZERO SUPPRESSION
*    PRICE     = ARITHMETIC, ZERO SUPPRESSION,
*                DECIMAL PLACES : 2, DIGIT GROUPING
*    TOTAL     = ARITHMETIC, ZERO SUPPRESSION,
*                DECIMAL PLACES : 2, DIGIT GROUPING
*    FUNCTION  = UPPERCASE LETTERS ONLY
*
*

 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
*  OTHER TRANSFER AREAS                                              *
*
*  FIELDS OF THE FORMCOB2 FORMAT                                     *
*  IN IFG, WHEN SPECIFYING THE ATTRIBUTE BLOCKS IN THE DATA TRANSFER
*  AREA, THE FOLLOWING ATTRIBUTE GROUPS MUST BE SET:
*    (BASIC ATTRIBUTES -  ALWAYS PRESENT)
*    FIELD INPUT       -  TO TEST THE INDIVIDUAL CHARACTERISTICS OF
```

```
*                        THE FIELDS, E.G. AN ENTERED FIELD WITH
*                        MANDATORY INPUT IS NOT REQUESTED AGAIN IN
*                        THE EVENT OF A DIFFERENCE OUTPUT.
*    DISPLAY CONTROL   -  TO CHANGE THE VISUAL APPEARANCE OF THE
*                        FIELDS, E.G. UNDERLINE
 01  FORMCOB2-TEXT.
     40  LENGTH2       PIC 9(5) COMP.
     COPY FORMCOB2 OF COBLIB.
* IFG COPY   NAME: FORMCOB2
*     FORMATNAME: FORMCOB2 USER AREA LENGTH : 00340   UTM TYPE: #
************************************************************
*                 GLOBAL ATTRIBUTE BLOCK                   *
************************************************************

    40 FORMCOB2-GLOBALS.
*        FORM-RETURNCODE
      41 RC-MAIN                      PIC 9(5) COMP SYNC.
      41 RC-CATEGORY                  PIC 9(4) COMP.
      41 RC-REASON                    PIC 9(4) COMP.
         FORM-INDICATORS
      41 FIELDS-MOD                   PIC X.
      41 FIELDS-DET                   PIC X.
      41 FIELDS-VALID                 PIC X.
      41 USER-EXIT-RC                 PIC X.
      41 FIELDS-UNDEFINED             PIC X.
         INPUT-IDENTIFICATION
      41 INPUT-KEY-CLASS              PIC X.
      41 INPUT-KEY-NUMBER             PIC 9(4) COMP.
      41 FILLER                       PIC X(4).
         DEVICE-CONTROLS
      41 INIT-CTL                     PIC X.
      41 INIT-OPT                     PIC X.
      41 TAB-CTL                      PIC X.
      41 FCT-LOCK                     PIC X.
      41 VMI-CTL                      PIC X.
      41 HMI-CTL                      PIC X.
      41 FILLER                       PIC X(2).
         OUTPUT-CONTROLS
      41 CYCLE-CTL                    PIC X.
      41 COPY-CTL                     PIC X.
      41 ALARM-CTL                    PIC X.
      41 HOLE-COLOR                   PIC X.
*        FORM-CONTROLS
      41 DISPLAY-SEL                  PIC X.
      41 LEVEL-SEL                    PIC X.
      41 OUTPUT-MODE                  PIC X.
      41 CURSOR-CTL                   PIC X.
```

```
      41 CURSOR-POS                    PIC 9(5) COMP.
      41 USER-EXIT-CTL                 PIC X.
      41 FILLER                        PIC X(3).
      41 STARTLINE                     PIC 9(4) COMP.
*
      41 P-KEY-SET                     PIC X(8).
*        DIALOG PARAM.        (attribute group DE-messages)
      41 MSG-IDENT                     PIC X(8).
      41 MSG-LOC                       PIC X(8).
      41 MSG-INDEX                     PIC 9(5) COMP.
*        DIALOG CURS. POS.    (attribute group cursor positioning)
      41 Z-CURSOR-FIELD                PIC X(8).
      41 Z-CURSOR-POS                  PIC 9(5) COMP.
      41 Z-CURSOR-INDEX                PIC 9(5) COMP.

************************************************************
*                 FIELD ATTRIBUTE BLOCKS                   *
************************************************************

    40 FORMCOB2-ATTR.
      41 CUSTNO-FAB.
        42 BASIC-ATTR.
          43 INPUT-STATE               PIC X.
          43 INPUT-STATE-ACT           PIC X.
          43 EDIT-STATE                PIC X.
          43 OUTPUT-CTL                PIC X.
        42 FIELD-INPUT.
          43 INPUT-CTL                 PIC X.
          43 PROTECTION                PIC X.
        42 DISPLAY-CTL.
          43 INTENSITY                 PIC X.
          43 VISIBILITY                PIC X.
          43 UNDERLINE                 PIC X.
          43 INVERSE                   PIC X.
        42 INIT-CURSOR                 PIC X.

      41 DATE-FAB.
        42 BASIC-ATTR.
          43 INPUT-STATE               PIC X.
          43 INPUT-STATE-ACT           PIC X.
          43 EDIT-STATE                PIC X.
          43 OUTPUT-CTL                PIC X.
        42 FIELD-INPUT.
          43 INPUT-CTL                 PIC X.
          43 PROTECTION                PIC X.
        42 DISPLAY-CTL.
          43 INTENSITY                 PIC X.
          43 VISIBILITY                PIC X.
```

```
    43 UNDERLINE                PIC X.
    43 INVERSE                  PIC X.
  42 INIT-CURSOR                PIC X.

41 ORDNO-FAB.
  42 BASIC-ATTR.
    43 INPUT-STATE              PIC X.
    43 INPUT-STATE-ACT          PIC X.
    43 EDIT-STATE               PIC X.
    43 OUTPUT-CTL               PIC X.
  42 FIELD-INPUT.
    43 INPUT-CTL                PIC X.
    43 PROTECTION               PIC X.
  42 DISPLAY-CTL.
    43 INTENSITY                PIC X.
    43 VISIBILITY               PIC X.
    43 UNDERLINE                PIC X.
    43 INVERSE                  PIC X.
  42 INIT-CURSOR                PIC X.

41 ITEM-FAB.
  42 BASIC-ATTR.
    43 INPUT-STATE              PIC X.
    43 INPUT-STATE-ACT          PIC X.
    43 EDIT-STATE               PIC X.
    43 OUTPUT-CTL               PIC X.
  42 FIELD-INPUT.
    43 INPUT-CTL                PIC X.
    43 PROTECTION               PIC X.
  42 DISPLAY-CTL.
    43 INTENSITY                PIC X.
    43 VISIBILITY               PIC X.
    43 UNDERLINE                PIC X.
    43 INVERSE                  PIC X.
  42 INIT-CURSOR                PIC X.

41 DESIG-FAB.
  42 BASIC-ATTR.
    43 INPUT-STATE              PIC X.
    43 INPUT-STATE-ACT          PIC X.
    43 EDIT-STATE               PIC X.
    43 OUTPUT-CTL               PIC X.


  42 FIELD-INPUT.
    43 INPUT-CTL                PIC X.
    43 PROTECTION               PIC X.
  42 DISPLAY-CTL.
    43 INTENSITY                PIC X.
```

```
    43 VISIBILITY              PIC X.
    43 UNDERLINE               PIC X.
    43 INVERSE                 PIC X.
  42 INIT-CURSOR               PIC X.

41 QTY-FAB.
  42 BASIC-ATTR.
    43 INPUT-STATE             PIC X.
    43 INPUT-STATE-ACT         PIC X.
    43 EDIT-STATE              PIC X.
    43 OUTPUT-CTL              PIC X.
  42 FIELD-INPUT.
    43 INPUT-CTL               PIC X.
    43 PROTECTION              PIC X.
  42 DISPLAY-CTL.
    43 INTENSITY               PIC X.
    43 VISIBILITY              PIC X.
    43 UNDERLINE               PIC X.
    43 INVERSE                 PIC X.
  42 INIT-CURSOR               PIC X.

41 PRICE-FAB.
  42 BASIC-ATTR.
    43 INPUT-STATE             PIC X.
    43 INPUT-STATE-ACT         PIC X.
    43 EDIT-STATE              PIC X.
    43 OUTPUT-CTL              PIC X.
  42 FIELD-INPUT.
    43 INPUT-CTL               PIC X.
    43 PROTECTION              PIC X.
  42 DISPLAY-CTL.
    43 INTENSITY               PIC X.
    43 VISIBILITY              PIC X.
    43 UNDERLINE               PIC X.
    43 INVERSE                 PIC X.
  42 INIT-CURSOR               PIC X.

41 TOTAL-FAB.
  42 BASIC-ATTR.
    43 INPUT-STATE             PIC X.
    43 INPUT-STATE-ACT         PIC X.
    43 EDIT-STATE              PIC X.
    43 OUTPUT-CTL              PIC X.
  42 FIELD-INPUT.
    43 INPUT-CTL               PIC X.
    43 PROTECTION              PIC X.
  42 DISPLAY-CTL.
    43 INTENSITY               PIC X.
```

```
          43 VISIBILITY              PIC X.
          43 UNDERLINE               PIC X.
          43 INVERSE                 PIC X.
        42 INIT-CURSOR               PIC X.

      41 FUNCTION-FAB.
        42 BASIC-ATTR.
          43 INPUT-STATE             PIC X.
          43 INPUT-STATE-ACT          PIC X.
          43 EDIT-STATE               PIC X.
          43 OUTPUT-CTL               PIC X.
        42 FIELD-INPUT.
          43 INPUT-CTL                PIC X.
          43 PROTECTION               PIC X.
        42 DISPLAY-CTL.
          43 INTENSITY                PIC X.
          43 VISIBILITY               PIC X.
          43 UNDERLINE                PIC X.
          43 INVERSE                  PIC X.
        42 INIT-CURSOR                PIC X.

      41 MESSLINE-FAB.
        42 BASIC-ATTR.
          43 INPUT-STATE             PIC X.
          43 INPUT-STATE-ACT          PIC X.
          43 EDIT-STATE               PIC X.
          43 OUTPUT-CTL               PIC X.
        42 FIELD-INPUT.
          43 INPUT-CTL                PIC X.
          43 PROTECTION               PIC X.
        42 DISPLAY-CTL.
          43 INTENSITY                PIC X.
          43 VISIBILITY               PIC X.
          43 UNDERLINE                PIC X.
          43 INVERSE                  PIC X.
        42 INIT-CURSOR                PIC x>

   40 FORMCOB2-ATTR-TAB REDEFINES FORMCOB2-ATTR OCCURS 010 TIMES.
        41 BASIC-ATTR.
          42 INPUT-STATE             PIC X.
          42 INPUT-STATE-ACT         PIC X.
          42 EDIT-STATE              PIC X.
          42 OUTPUT-CTL              PIC X.
        41 FIELD-INPUT.
          42 INPUT-CTL               PIC X.
          42 PROTECTION              PIC X.
        41 DISPLAY-CTL.
```

```
        42 INTENSITY                PIC X.
        42 VISIBILITY               PIC X.
        42 UNDERLINE                PIC X.
        42 INVERSE                  PIC X.
      41 INIT-CURSOR                PIC X.


*************************************************************
*                    FIELD DATA PART                        *
*************************************************************

    40 FORMCOB2-DATA.
      41 CUSTNO                          PIC X(014).
      41 DATE                            PIC X(014).
      41 ORDNO                           PIC X(012).
      41 ITEM                            PIC X(003).
      41 DESIG                           PIC X(039).
      41 QTY                             PIC  9(003).
      41 PRICE                           PIC  9(004)V9(002).
      41 TOTAL                           PIC  9(007)V9(002).
      41 FUNCTION                        PIC X(008).
      41 MESSLINE                        PIC X(080).
*
*  CONTROL-AREAS-----------------------------------------------*
 01  FHS-AREAS1.
     COPY FHSMAINP OF FHSLIB.
*                       ^   = LINKNAME FOR COBOL LIBRARY
*                         CONTAINING FHS DATA STRUCTURES
*                         DEFAULT LIBRARY: SYSLIB.FHS
*****************************************************************
*                                                               *
*    FHSMAINP version 800                                       *
*                                                               *
*                      /->  FHS-CONTROL-INFO                    *
*       FHS-MAIN-PAR -                                          *
*                      \->  FHS-MAP-PAR                         *
*****************************************************************
     .
     .
     .    FOR EXPLANATION AND COMPLETE OUTPUT OF THIS
     .    COPY ELEMENT SEE UNDER 'FHS-MAIN-PAR' IN THE MANUAL

     .
     .
     .
*
     COPY TIAMINFO OF TIAMLIB.
*                       ^   = LINKNAME FOR TIAM LIBRARY
```

```
*                              CONTAINING FHS DATA STRUCTURES
*                              DEFAULT LIBRARY: SYSLIB.TIAM
*************************************************************
*      TIAMINFO   002         920620        TIAM      U     *
*************************************************************
     .
     .
     .    FOR EXPLANATION AND COMPLETE OUTPUT OF THIS
     .    COPY ELEMENT SEE THE TIAM MANUAL
     .
     .
     .
*
     COPY FHSAVAL OF FHSLIB.
*                       ^   = LINKNAME FOR COBOL LIBRARY
*                             CONTAINING FHS DATA STRUCTURES
*                             DEFAULT LIBRARY: SYSLIB.FHS
*****************************************************************
*    FHSAVAL  version 800                                       *
*                                                               *
*    ATTRIBUTE VALUES FOR IFG ADDRESSING AID                    *
*                                                               *
*****************************************************************
 01  FHS-ATTRIBUTE-VALUES.
*****************************************************************
*
*   GLOBAL ATTRIBUTE VALUES (CHARACTERS)
*
*    GA-DEFAULT-VALUES.
           02  GA-DEFAULT             PIC X VALUE SPACE.
*    FORMATTING-INDICATORS.
*       FIELDS-MODIFICATION.
           02  GA-MODIFIED            PIC X VALUE "Y".
           02  GA-NOT-MODIFIED        PIC X VALUE " ".
*       FIELDS-DETECTION.
           02  GA-DETECTED            PIC X VALUE "Y".
           02  GA-NOT-DETECTED        PIC X VALUE " ".
*       FIELDS-VALIDATION.
           02  GA-VALID               PIC X VALUE "V".
           02  GA-NOT-VALID           PIC X VALUE " ".
*       FIELDS-UNDEFINED.
           02  GA-UNDEFINED           PIC X VALUE "Y".
           02  GA-NOT-UNDEFINED       PIC X VALUE " ".
*    INPUT-IDENTIFICATION.
*       INPUT-KEY-CLASS.
           02  GA-INPUT-KEY           PIC X VALUE "I".
           02  GA-F-KEY               PIC X VALUE "F".
```

```
           02  GA-K-KEY              PIC X VALUE "K".
           02  GA-POS-RM             PIC X VALUE "P".
           02  GA-NEG-RM             PIC X VALUE "N".
           02  GA-INPUT-NONE         PIC X VALUE " ".
*    DEVICE-CONTROLS.
*       INIT-CONTROL.
           02  GA-NO-INIT            PIC X VALUE "N".
           02  GA-FIRST-INIT         PIC X VALUE "F".
           02  GA-LAST-INIT          PIC X VALUE "L".
           02  GA-BOTH-INIT          PIC X VALUE "B".
*       TABULATOR-CONTROL.
           02  GA-AUTO-TAB           PIC X VALUE "A".
           02  GA-NO-AUTO-TAB        PIC X VALUE "N".
*       FUNCTION-LOCK.
           02  GA-KEYLOCK            PIC X VALUE "K".
*       VMI-CONTROL.
           02  GA-VMI-1              PIC X VALUE "1".
           02  GA-VMI-2              PIC X VALUE "2".
           02  GA-VMI-3              PIC X VALUE "3".
*       HMI-CONTROL.
           02  GA-HMI-1              PIC X VALUE "1".
           02  GA-HMI-2              PIC X VALUE "2".
           02  GA-HMI-3              PIC X VALUE "3".
*    OUTPUT-CONTROLS.
*       CYCLE-CONTROL.
           02  GA-CLOSE              PIC X VALUE "C".
*       COPY-CONTROL.
           02  GA-HARDCOPY-GEN       PIC X VALUE "H".
           02  GA-HARDCOPY-LOC       PIC X VALUE "L".
*       ALARM-CONTROL.
           02  GA-ALARM              PIC X VALUE "A".
*       HOLE-COLOR.
           02  GA-NO-COLOR           PIC X VALUE "U".
           02  GA-GREY-HOLE          PIC X VALUE "G".
           02  GA-WHITE-HOLE         PIC X VALUE "W".
*    FORMATTING-CONTROLS.
*       DISPLAY-SELECTION.
           02  GA-BOXB               PIC X VALUE "B".
           02  GA-BOXL               PIC X VALUE "C".
           02  GA-KEB                PIC X VALUE "K".
           02  GA-KEL                PIC X VALUE "L".
*       LEVEL-SELECTION.
           02  GA-LEVEL-1            PIC X VALUE "1".
           02  GA-LEVEL-2            PIC X VALUE "2".
           02  GA-LEVEL-3            PIC X VALUE "3".
           02  GA-LEVEL-P            PIC X VALUE "P".
*       OUTPUT-MODE.
           02  GA-RDIF               PIC X VALUE "R".
```

```
*       CURSOR-CONTROL.
           02  GA-FIELD-CURSOR        PIC X VALUE "F".
           02  GA-EDIT-CURSOR         PIC X VALUE "E".
           02  GA-REL-CURSOR          PIC X VALUE "R".
*       USER-EXIT-CONTROL.
           02  GA-NO-UEXIT            PIC X VALUE "N".
           02  GA-OUT-UEXIT           PIC X VALUE "O".
           02  GA-IN-UEXIT            PIC X VALUE "I".
           02  GA-BOTH-UEXIT          PIC X VALUE "B".


*   FIELD ATTRIBUTE VALUES (CHARACTERS)
*
*    DEFAULT-VALUES.
           02  FA-DEFAULT             PIC X VALUE SPACE.
*
*    BASIC-ATTRIBUTES.
*       INPUT-STATE-/-INPUT-STATE-ACT.
           02  FA-MODIFIED            PIC X VALUE "M".
           02  FA-CLEARED             PIC X VALUE "C".
           02  FA-DETECTED            PIC X VALUE "D".
           02  FA-UNDEFINED           PIC X VALUE "U".
           02  FA-NOT-TOUCHED         PIC X VALUE " ".
*       EDIT-STATE.
           02  FA-VALID               PIC X VALUE "V".
           02  FA-INVALID             PIC X VALUE "I".
           02  FA-MUST-ERROR          PIC X VALUE "M".
           02  FA-NOT-CHECKED         PIC X VALUE " ".
*       OUTPUT-CONTROL.
           02  FA-OUTPUT-INIT         PIC X VALUE "I".
           02  FA-OUTPUT-DATA         PIC X VALUE "D".
           02  FA-OUTPUT-UNDEFINED    PIC X VALUE "U".
*
*    FIELD-INPUT.
*       INPUT-CONTROL.
           02  FA-NORMAL-IN           PIC X VALUE "N".
           02  FA-MUST-IN             PIC X VALUE "M".
           02  FA-POTMUST-IN          PIC X VALUE "P".
           02  FA-AUTORET-IN          PIC X VALUE "A".
*       PROTECTION.
           02  FA-UNPROTECTED         PIC X VALUE "U".
           02  FA-PROTECTED           PIC X VALUE "P".
           02  FA-ASKIP               PIC X VALUE "A".
           02  FA-DETECTABLE          PIC X VALUE "D".
*
*    DISPLAY-CONTROL.
*       INTENSITY.
           02  FA-HIGH-INTENSITY      PIC X VALUE "H".
```

```
           02  FA-NORMAL-INTENSITY    PIC X VALUE "N".
*       VISIBILITY.
           02  FA-VISIBLE             PIC X VALUE "V".
           02  FA-SIGNALING           PIC X VALUE "S".
           02  FA-INVISIBLE           PIC X VALUE "I".
*       UNDERLINE.
           02  FA-UNDERLINED          PIC X VALUE "Y".
           02  FA-NOT-UNDERLINED      PIC X VALUE "N".
*       INVERSE.
           02  FA-INVERSE             PIC X VALUE "Y".
           02  FA-NOT-INVERSE         PIC X VALUE "N".
*
*    COLOUR.
           02  FA-RED                 PIC X VALUE "1".
           02  FA-GREEN               PIC X VALUE "2".
           02  FA-YELLOW              PIC X VALUE "3".
           02  FA-BLUE                PIC X VALUE "4".
           02  FA-MAGENTA             PIC X VALUE "5".
           02  FA-CYAN                PIC X VALUE "6".
           02  FA-WHITE               PIC X VALUE "7".
           02  FA-NO-COLOUR           PIC X VALUE "N".
*
*    CURSOR.
           02  FA-CURSOR              PIC X VALUE "Y".
           02  FA-HOLD-CURSOR         PIC X VALUE "H".
           02  FA-NO-CURSOR           PIC X VALUE "N".
*****************************************************************
*
*  AUXILIARY FIELD-------------------------------------------------*
 01  NEW              PIC 9.
*
*
*****************************************************************
*  PROGRAM START                                                 *
*****************************************************************
*
*
 PROCEDURE DIVISION.
 CONTR SECTION.
 CONTR-0.
     PERFORM MAIN
             UNTIL FUNCTION = "END".
 CONTR-9.
     DISPLAY "                    H E L L O !!!" UPON TERMINAL.
     STOP RUN.
*
```

```
*****************************************************************
*
 MAIN SECTION.
 MAIN-0.
     PERFORM PREPARATION.
     PERFORM OUTPUT1.
 MAIN-9.
     EXIT.
*
*****************************************************************
*
 PREPARATION SECTION.
 PREPARATION-0.
*
*  SET FHS-MODE IN FIELD "EDIT-MODE"                            *
*
     MOVE "F" TO EDIT-MODE IN EDIT-OUT IN TIAM-CONTROL-INFO.
*
*  ASSIGN OWN FORMAT APPLICATION FILE                           *
*
     MOVE "Y" TO FHS-MAP-LIB-OPT.
     MOVE "$C.PROGRAMEXAMPLES.LMSLIB" TO FHS-MAP-LIB-NAME.
*
*  #FORMATS REQUIRE A RESTART AREA                              *
*
     MOVE "Y" TO FHS-RESTART-OPT1.
*
*  INITIALIZATION AT START OF OUTPUT                            *
*
     MOVE GA-FIRST-INIT TO INIT-CTL IN FORMCOB2-GLOBALS.
*
*  NAME OF FORMAT TO BE FORMATTED                               *
*
     MOVE "FORMCOB2" TO FHS-MAP-NAME.
*
*  DELETE DATA TRANSFER AREA                                    *
*
     MOVE LOW-VALUE TO FORMCOB2-DATA.
*
*  NEW INPUT OF FORMAT                                          *
*
     MOVE ZERO TO NEW.
 PREPARATION-9.
     EXIT.
*
*****************************************************************
*
 OUTPUT1 SECTION.
```

```
 OUTPUT-O.
*
*  OUTPUT/INPUT OF FORMAT                                          *
*
     CALL "WRTRD" USING TIAM-CONTROL-INFO
                        FORMCOB2-TEXT
                        FORMCOB2-TEXT
                        FHS-MAIN-PAR.
*
*  TECHNICAL ERROR ?
*  IF YES, ISSUE RETURN CODE AND TERMINATE PROGRAM                 *
*
     IF TIAM-RC NOT ZERO
        THEN PERFORM ERRORROUTINE.
*
*  ERROR ON INPUT ?
*  IF YES, NO INITIALIZATION AT START OF OUTPUT
*          POSITION CURSOR TO 1ST FIELD CONTAINING EDIT ERROR
*  IF NO, PROCESSING OF FIELDS                                     *
*
     IF FIELDS-VALID OF FORMCOB2-GLOBALS NOT = GA-VALID
        THEN MOVE GA-NO-INIT TO INIT-CTL IN FORMCOB2-GLOBALS
             MOVE GA-EDIT-CURSOR TO CURSOR-CTL IN FORMCOB2-GLOBALS
        ELSE GO TO OUTPUT-5.
*
*  FIELDS WITH MANDATORY INPUT ENTERED / DATE CORRECT ?
*  IF NO, ISSUE MESSAGE                                            *
*
     IF EDIT-STATE IN CUSTNO-FAB = FA-MUST-ERROR
        THEN MOVE "CUSTOMER NO. NOT YET ENTERED"
                  TO MESSLINE
             GO TO OUTPUT-7.
*
     IF EDIT-STATE IN DATE-FAB NOT = FA-VALID
        THEN MOVE "DATE (DD.MM.YYYY) ENTERED INCORRECTLY"
                  TO MESSLINE
             GO TO OUTPUT-7.
*
     IF EDIT-STATE IN ORDNO-FAB = FA-MUST-ERROR
        THEN MOVE "ORDER NO. NOT YET ENTERED"
                  TO MESSLINE
             GO TO OUTPUT-7.
*
*  FIELD WITH MANDATORY INPUT ENTERED ?
*  IF NO, UNDERLINE POSITION AND ISSUE MESSAGE                     *
*
```

```
 IF EDIT-STATE IN ITEM-FAB = FA-MUST-ERROR
        THEN MOVE FA-UNDERLINED TO UNDERLINE IN ITEM-FAB
                                   UNDERLINE IN DESIG-FAB
                                   UNDERLINE IN QTY-FAB
                                   UNDERLINE IN PRICE-FAB
             MOVE "NOT ALL FIELDS HAVE BEEN ENTERED!"
                  TO MESSLINE
             GO TO OUTPUT-7.
*
*  FIELD WITH MANDATORY INPUT ENTERED / MINIMUM INPUT ?
*  IF NO, ISSUE MESSAGE                                          *
*
     IF EDIT-STATE IN DESIG-FAB = FA-MUST-ERROR
        THEN MOVE "DESIGNATION NOT YET ENTERED"
                  TO MESSLINE
        ELSE IF EDIT-STATE IN DESIG-FAB = FA-INVALID
                THEN MOVE "MINIMUM INPUT FOR DESIG. = 4 CHARS."
                     TO MESSLINE.
     GO TO OUTPUT-7.
*
 OUTPUT-5.
*
*  CALCULATE TOTAL                                               *
*
     IF NEW = 1
        THEN GO TO OUTPUT-9.
     MULTIPLY QTY BY PRICE
              GIVING TOTAL.
     MOVE SPACE TO MESSLINE.
     MOVE 1 TO NEW.
*
*  OUTPUT POSITION NOT UNDERSCORED                               *
*
     MOVE FA-NOT-UNDERLINED TO UNDERLINE IN ITEM-FAB
                               UNDERLINE IN DESIG-FAB
                               UNDERLINE IN QTY-FAB
                               UNDERLINE IN PRICE-FAB.
*
 OUTPUT-7.
*
*  DIFFERENCE OUTPUT OF FORMAT                                   *
*
     GO TO OUTPUT-0.
*
 OUTPUT-9.
     EXIT.
*
```

```
*****************************************************************
*
 ERRORROUTINE SECTION.
 ERROR-0.
     IF TIAM-RC NOT ZERO
        THEN DISPLAY "INPUT/OUTPUT ERROR IN " FHS-MAP-NAME "!   "
-            "RETURNCODE: " TIAM-RC UPON TERMINAL.
     IF FHS-MAIN-RC NOT ZERO
        THEN DISPLAY "FORMATTING ERROR IN " FHS-MAP-NAME "
-            "!    RETURNCODE: " ERROR-CATEGORY " " ERROR-REASON
             UPON TERMINAL.
     GO TO CONTROL-9.
 ERROR-9.
     EXIT.
*
*****************************************************************
```

8 FHS in Fortran programs

This chapter contains all you need to know if you wish to use FHS in Fortran programs with the TIAM access method. The required Fortran data structures are listed starting on page 423.

8.1 Structure of FHS Fortran programs

The FHS Fortran interface enables the Fortran programmer to implement FHS functions in TIAM application programs without having to write an ASSEMBLER subprogram for the formatting. The formatting functions have been integrated into the calls of the access method. You merely have to supply certain data structures with data before issuing the input/output call. FHS uses these structures to obtain formatting parameters and to store return codes. Formatting is possible using the following TIAM calls (see page 440f):

– CALL “WROUT”
– CALL “WRTRD”

In addition, you can make use of the following FHS Fortran calls, implemented in the form of subprogram calls (CALL ...):

– CALL “FHSCURS” for exact positioning of the cursor in +formats and in *formats (see page 442).
– CALL “FHSATTR” for modifying field attributes in +formats (see page 442).
– CALL “FHSINIT” for initializing the formatting and defining certain start parameters for formatting (see page 443).
– CALL “FHSSERV” for calling special FHS service functions (see page 444).

Attributes can be modified with the aid of the include member FFOAVAL which contains a complete list of the symbolic attribute names.

You generate the formats using IFG.

The data structures used by FHS Fortran are provided in the form of include members. This simplifies the transfer of the formatting parameters to FHS. These data structures are described in the section starting on page 423. Points to be observed when compiling and linking FHS Fortran programs are given in the section starting on page 453.

FHS Fortran programs have the following structure:

```
      PROGRAM XXXXXX                          program name
*
      %INCLUDE FFOMAINP                       ready FHS-MAIN-PAR
     [%INCLUDE FFOATTRP]                      ready FHS-ATTR-PAR
     [%INCLUDE FFOATTRM]                      include member for ATTRIBUTE-MOVE
     [%INCLUDE FFOINITP]                      ready FHS-INIT-PAR
     [%INCLUDE FFOAVAL]                       ready FHS-ATTRIBUTE-VALUES
     [%INCLUDE FFOEXITP,'EXIT'=EXITF']        ready FHS-EXITMOD-PAR
     [%INCLUDE FFOCCSNP]                      ready FHS-CCSN-PAR
     [%INCLUDE TIFOINFO]                      TIAM-CONTROL-INFO
*
      EXTERNAL YYYYYY                         name of TIAM call
*
     [CALL FHSINIT(...)]                      define start parameters
*
        .
        .
        .
     [CALL FHSCURS(...);]                     position CURSOR
        .
        .
      CALL YYYYYY(...)                        TIAM call(s)
        .
     [attribute modifications using FFOAVAL]
        .
*
```

Optional entries appear in brackets. The include members are stored in the FHS library SYSLIB.FHS... .

8.2 Data structures used by FHS Fortran

Data structures provide the interface between FHS and the application program. They are stored as include members in the FHS library (SYSLIB.FHS.083) from where they are copied into the application program. In the Fortran program, they are specified in the CALL macro.

Except for the omitted hyphens, the field names of the Fortran data structures are identical with the COBOL field names. You are therefore referred to the description of COBOL data structures.

The following data structures are available:

FHSMAINPAR

This data structure is divided into two parts. In “FHSCONTROLINFO”, the application program is supplied with information about the formatting run (return codes etc.). “FHSMAPPAR” is used by the application program to control formatting.
The FHSMAINPAR data structure is copied into the program using the Fortran statement “**%INCLUDE FFOMAINP**”. FHSMAINPAR is shown on page 425. For an explanation of the meaning of the individual fields please refer to section “The FHS-MAIN-PAR data structure” on page 303.

FHSINITPAR

This data structure is used in the FHSINIT call to supply application program-specific default values for subsequent formatting operations and to specify the format application file. The FHSINITPAR data structure is copied into the application program by means of the Fortran statement “**%INCLUDE FFOINITP**”. FHSINITPAR is shown on page 431. For an explanation of the meaning of the individual fields please refer to section “The FHS-INIT-PAR data structure” on page 330.

FHSATTRPAR

This data structure is required to modify attributes of +formats using the FHSATTR call.

This data structure is copied into the application program by means of the Fortran statement “**%INCLUDE FFOATTRP**”. FHSATTRPAR is shown on page 435.

FHSEXITMODPAR

This data structure corresponds to the user exit interface and is only required when exit routines are used (see section “Checking data fields with an exit routine” on page 41. FHSEXITMODPAR is copied into the program by means of the Fortran statement “**%INCLUDE FFOEXITP,’EXIT’=’EXITF**’”. FHSEXITMODPAR is shown on page 437. FHSEXITMODPAR is listed on page 437. For an explanation of the meaning of the individual fields refer to the section on “The FHS-EXIT-MOD-PAR data structure”, in the COBOL chapter.

FHSCCSNPAR

The FHSSERV call requires this data structure to obtain the character set used in the format. It is copied into the application program by means of the Fortran statement “**%INCLUDE FFOCCSNP**”.

FHSATTRIBUTEVALUES

This list generates symbolically addressable attribute values for the global attributes and field attributes of the data transfer area with separate attribute blocks and field contents. It is copied into the application program by means of the Fortran statement “**%INCLUDE FFOAVAL**”. FHSATTRIBUTEVALUES is shown on page 446.

FHSATTRIBUTEMOVE

This data structure is required to modify attributes of +formats by means of a statement (instead of a FHSATTR call).

This data structure is copied into the program by means of the Fortran statement “**%INCLUDE FFOATTRM**”. INCLUDE FFOATTRM is shown on page 450.

The table on page 302 shows which data structures are required for which FHS Fortran calls.

8.2.1 The FHSMAINPAR data structure

FHSMAINPAR is the name of the data structure below which is copied into the program by means of "**%INCLUDE FFOMAINP**".

```
************************************************************************
* NAME              FFOMAINP
* VERSION           811
*
*                    /->  FHS-CONTROL-INFO
*     FHS-MAIN-PAR -
*                    \->  FHS-MAP-PAR
*
************************************************************************
*
      INTEGER*4     FHSMAINDUMMY
*                                              FHS MAIN PAR
      CHARACTER*384 FHSMAINPAR
      EQUIVALENCE  (FHSMAINDUMMY,FHSMAINPAR(1:4))
*
*                                             FHS CONTROL INFO
      CHARACTER*56  FHSCONTROLINFO
*                                               FHS MAIN RC
      INTEGER*2     FHSMAINRC
*                                               FHS ERROR INFO
      CHARACTER*4   FHSERRORINFO
*                                                 ERROR CATEGORY
      INTEGER*2     ERRORCATEGORY
*                                                 ERROR REASON
      INTEGER*2     ERRORREASON
*                                               PRINTER RETURN MSG
      CHARACTER*5   PRINTRETURNMSG
*                                                 RETURN MSG TYPE
      CHARACTER*1   RETURNMSGTYPE
*                                                 RETURN BYTE1
      CHARACTER*1   RETURNBYTE1
*                                                 RETURN BYTE2
      CHARACTER*1   RETURNBYTE2
*                                                 RETURN STATUS INFO
      CHARACTER*2   RETURNSTATINFO
*                                               FHS OUTPUT INFO
      CHARACTER*16  FHSOUTPUTINFO
*                                                 OUT USER AREA TRUNCATION
      CHARACTER*1   OUTUSERAREATRUN
*                                                 OUT USER AREA LEN
      INTEGER*4     OUTUSERAREALEN
*                                               FHS INPUT INFO
```

```
      CHARACTER*16  FHSINPUTINFO
*
      CHARACTER*1   INPRINTERRETMSG
*
      CHARACTER*1   INFIELDDET
*
      CHARACTER*1   INMSGNILS
*
      INTEGER*2     INFKEY
*
      INTEGER*2     INKKEY
*
      INTEGER*4     INUSERAREALEN
*
      INTEGER*4     INMSGLEN
*
*
      CHARACTER*328 FHSMAPPAR
*
      CHARACTER*160 FHSMAPPARGEN
*
      CHARACTER*160 FHSMAPGENERALS
*
      CHARACTER*8   FHSMAPNAME
*
      CHARACTER*8   FHSEXITMODNAME
*
      CHARACTER*4   FHSMAPPINGMETH
*
      CHARACTER*1   FHSMODYATTRS
*
      CHARACTER*1   FHSPARTMAPOPT
*
      CHARACTER*1   FHSMAPPART
*
      CHARACTER*1   FHSMAPCURSOROPT
*
      INTEGER*2     FHSSERVICEFUNCT
*
      CHARACTER*1   FHSRESTARTOPT1
*
      CHARACTER*1   FHSRESTARTOPT2
*
      CHARACTER*1   FHSMAPLIBOPT
*
      CHARACTER*54  FHSMAPLIBNAME
*
      CHARACTER*1   FHSEXITLIBOPT
```

```
    IN PRINTER RETURN MSG

    IN FIELD DET

    IN MSG NILS

    IN F-KEY

    IN K-KEY

    IN USER AREA LEN

    IN MSG LEN


FHS MAP PAR

  FHS MAP PAR GENERAL

  FHS MAP GENERALS

    FHS MAP NAME

    FHS EXIT MOD NAME

    FHS MAPPING METHOD

    FHS MODY ATTRS

    FHS PARTIAL MAP OPT

    FHS MAP PART

    FHS MAP CURSOR OPT

    FHS SERVICE FUNCTION

    FHS RESTART OPT1

    FHS RESTART OPT2

    FHS MAP LIB OPT

    FHS MAP LIB NAME

    FHS EXIT LIB OPT
```

```
*
      CHARACTER*54  FHSEXITLIBNAME
*
      CHARACTER*1   FHSEXITFOROUT
*
      CHARACTER*1   FHSEXITFORIN
*
      CHARACTER*8   FHSDESIREDCCSN
*
      CHARACTER*60  FHSMAPPAROPTION
*
      CHARACTER*60  FHSMAPOPTIONS
*
      CHARACTER*4   MAPDEVICECLASS
*
      CHARACTER*4   MAPPRINTERCONTR
*
      CHARACTER*1   MAPAUTOTAB
*
      CHARACTER*1   MAPEFFLEN
*
      CHARACTER*1   MAPPOSDETCHAR
*
      CHARACTER*1   MAPNEGDETCHAR
*
      CHARACTER*4   MAPREADMETHOD
*
      CHARACTER*1   MAPSCREENPREMOD
*
      CHARACTER*1   MAPREADNILS
*
      CHARACTER*1   MAPUSEALLATTRS
*
      CHARACTER*1   MAPPRTOPTION
*
      CHARACTER*1   MAPPRTRETBYTE1
*
      CHARACTER*1   MAPPRTRETBYTE2
*
      CHARACTER*1   MAPHARDCOPYOPT
*
      CHARACTER*1   MAPAUTOHARDCOPY
*
      CHARACTER*1   MAPLOCKKEYS
*
      CHARACTER*1   MAPCLEAROPTION
*
      CHARACTER*1   MAPBELOPTION
```

```
  FHS EXIT LIB NAME

  FHS EXIT FOR OUTPUT

  FHS EXIT FOR INPUT

  FHS DESIRED CCSNAME

FHS MAP PAR OPTIONAL

FHS MAP OPTIONS

  MAP DEVICE CLASS

  MAP PRINTER CONTROL

  MAP AUTO TAB

  MAP EFF LEN

  MAP POS DET CHAR

  MAP NEG DET CHAR

  MAP READ METHOD

  MAP SCREEN PRE MOD

  MAP READ NILS

  MAP USE ALL ATTRS

  MAP PRINTER OPTION

  MAP PRINTER RETURN BYTE1

  MAP PRINTER RETURN BYTE2

  MAP HARDCOPY OPTION

  MAP AUTO HARDCOPY

  MAP LOCK KEYS

  MAP CLEAR OPTION

  MAP BEL OPTION
```

```
*                                                    MAP PRINT FORMAT OPTION
      CHARACTER*4   MAPPRTFORMATOPT
*                                                      MAP PRINT LINES
      CHARACTER*1   MAPPRINTLINES
*                                                      MAP PRINT COLUMNS
      CHARACTER*1   MAPPRINTCOLUMNS
*                                                      MAP PRINT PAPER
      CHARACTER*1   MAPPRINTPAPER
*                                                      MAP PRINT FORM
      CHARACTER*1   MAPPRINTFORM
*                                                    MAP LIB LOAD OPTION
      CHARACTER*2   MAPLIBLOADOPT
*                                                      MAP LIB LOAD MODE
      CHARACTER*1   MAPLIBLOADMODE
*                                                      MAP LIB LOAD FILE
      CHARACTER*1   MAPLIBLOADFILE
*                                                      MAP HOLE COLOR
      CHARACTER*1   MAPHOLECOLOR
*                                                  FHS EXIT PAR
      CHARACTER*108 FHSEXITPAR
*                                                    EXIT IDENT LEN
      INTEGER*4     EXITIDENTLEN
*                                                    EXIT IDENT
      CHARACTER*8   EXITIDENT
*                                                    EXIT IN OUT
      CHARACTER*1   EXITINOUT
*                                                    EXIT RET INFO
      CHARACTER*1   EXITRETINFO
*                                                    EXIT FLD LEN
      INTEGER*4     EXITFLDLEN
*                                                    EXIT EFF LEN
      INTEGER*4     EXITEFFLEN
*                                                    EXIT DATA
      CHARACTER*80  EXITDATA
*
*****************************************************************
*
      EQUIVALENCE (OUTUSERAREALEN, FHSOUTPUTINFO ( 13: 16))
      EQUIVALENCE (FHSMAINPAR      (  1: 56), FHSCONTROLINFO)
      EQUIVALENCE (FHSCONTROLINFO  (  1:  2), FHSMAINRC)
      EQUIVALENCE (FHSCONTROLINFO  (  9: 12), FHSERRORINFO)
      EQUIVALENCE (FHSERRORINFO    (  1:  2), ERRORCATEGORY)
      EQUIVALENCE (FHSERRORINFO    (  3:  4), ERRORREASON)
      EQUIVALENCE (FHSCONTROLINFO  ( 20: 24), PRINTRETURNMSG)
      EQUIVALENCE (PRINTRETURNMSG  (  1:  1), RETURNMSGTYPE)
      EQUIVALENCE (PRINTRETURNMSG  (  2:  2), RETURNBYTE1)
      EQUIVALENCE (PRINTRETURNMSG  (  3:  3), RETURNBYTE2)
      EQUIVALENCE (PRINTRETURNMSG  (  4:  5), RETURNSTATINFO)
```

```
EQUIVALENCE (FHSCONTROLINFO  ( 25: 40), FHSOUTPUTINFO)
EQUIVALENCE (FHSOUTPUTINFO   ( 12: 12), OUTUSERAREATRUN)
EQUIVALENCE (FHSOUTPUTINFO   ( 13: 16), OUTUSERAREALEN)
EQUIVALENCE (FHSCONTROLINFO  ( 41: 56), FHSINPUTINFO)
EQUIVALENCE (FHSINPUTINFO    (  2:  2), INPRINTERRETMSG)
EQUIVALENCE (FHSINPUTINFO    (  3:  3), INFIELDDET)
EQUIVALENCE (FHSINPUTINFO    (  4:  4), INMSGNILS)
EQUIVALENCE (FHSINPUTINFO    (  5:  6), INFKEY)
EQUIVALENCE (FHSINPUTINFO    (  7:  8), INKKEY)
EQUIVALENCE (FHSINPUTINFO    (  9: 12), INUSERAREALEN)
EQUIVALENCE (FHSINPUTINFO    ( 13: 16), INMSGLEN)
EQUIVALENCE (FHSMAINPAR      ( 57:384), FHSMAPPAR)
EQUIVALENCE (FHSMAPPAR       (  1:160), FHSMAPPARGEN)
EQUIVALENCE (FHSMAPPARGEN    (  1:160), FHSMAPGENERALS)
EQUIVALENCE (FHSMAPGENERALS  (  1:  8), FHSMAPNAME)
EQUIVALENCE (FHSMAPGENERALS  (  9: 16), FHSEXITMODNAME)
EQUIVALENCE (FHSMAPGENERALS  ( 17: 20), FHSMAPPINGMETH)
EQUIVALENCE (FHSMAPGENERALS  ( 21: 21), FHSMODYATTRS)
EQUIVALENCE (FHSMAPGENERALS  ( 22: 22), FHSPARTMAPOPT)
EQUIVALENCE (FHSMAPGENERALS  ( 23: 23), FHSMAPPART)
EQUIVALENCE (FHSMAPGENERALS  ( 24: 24), FHSMAPCURSOROPT)
EQUIVALENCE (FHSMAPGENERALS  ( 33: 34), FHSSERVICEFUNCT)
EQUIVALENCE (FHSMAPGENERALS  ( 35: 35), FHSRESTARTOPT1)
EQUIVALENCE (FHSMAPGENERALS  ( 36: 36), FHSRESTARTOPT2)
EQUIVALENCE (FHSMAPGENERALS  ( 37: 37), FHSMAPLIBOPT)
EQUIVALENCE (FHSMAPGENERALS  ( 38: 91), FHSMAPLIBNAME)
EQUIVALENCE (FHSMAPGENERALS  ( 92: 92), FHSEXITLIBOPT)
EQUIVALENCE (FHSMAPGENERALS  ( 93:146), FHSEXITLIBNAME)
EQUIVALENCE (FHSMAPGENERALS  (147:147), FHSEXITFOROUT)
EQUIVALENCE (FHSMAPGENERALS  (148:148), FHSEXITFORIN)
EQUIVALENCE (FHSMAPGENERALS  (149:156), FHSDESIREDCCSN)
EQUIVALENCE (FHSMAPPAR       (161:220), FHSMAPPAROPTION)
EQUIVALENCE (FHSMAPPAROPTION (  1: 60), FHSMAPOPTIONS)
EQUIVALENCE (FHSMAPOPTIONS   (  1:  4), MAPDEVICECLASS)
EQUIVALENCE (FHSMAPOPTIONS   (  5:  8), MAPPRINTERCONTR)
EQUIVALENCE (FHSMAPOPTIONS   ( 13: 13), MAPAUTOTAB)
EQUIVALENCE (FHSMAPOPTIONS   ( 14: 14), MAPEFFLEN)
EQUIVALENCE (FHSMAPOPTIONS   ( 15: 15), MAPPOSDETCHAR)
EQUIVALENCE (FHSMAPOPTIONS   ( 16: 16), MAPNEGDETCHAR)
EQUIVALENCE (FHSMAPOPTIONS   ( 25: 28), MAPREADMETHOD)
EQUIVALENCE (FHSMAPOPTIONS   ( 29: 29), MAPSCREENPREMOD)
EQUIVALENCE (FHSMAPOPTIONS   ( 30: 30), MAPREADNILS)
EQUIVALENCE (FHSMAPOPTIONS   ( 31: 31), MAPUSEALLATTRS)
EQUIVALENCE (FHSMAPOPTIONS   ( 32: 32), MAPPRTOPTION)
EQUIVALENCE (FHSMAPOPTIONS   ( 33: 33), MAPPRTRETBYTE1)
EQUIVALENCE (FHSMAPOPTIONS   ( 34: 34), MAPPRTRETBYTE2)
EQUIVALENCE (FHSMAPOPTIONS   ( 36: 36), MAPHARDCOPYOPT)
EQUIVALENCE (FHSMAPOPTIONS   ( 39: 39), MAPAUTOHARDCOPY)
```

```
      EQUIVALENCE (FHSMAPOPTIONS   ( 40: 40), MAPLOCKKEYS)
      EQUIVALENCE (FHSMAPOPTIONS   ( 41: 41), MAPCLEAROPTION)
      EQUIVALENCE (FHSMAPOPTIONS   ( 42: 42), MAPBELOPTION)
      EQUIVALENCE (FHSMAPOPTIONS   ( 43: 46), MAPPRTFORMATOPT)
      EQUIVALENCE (MAPPRTFORMATOPT (  1:  1), MAPPRINTLINES)
      EQUIVALENCE (MAPPRTFORMATOPT (  2:  2), MAPPRINTCOLUMNS)
      EQUIVALENCE (MAPPRTFORMATOPT (  3:  3), MAPPRINTPAPER)
      EQUIVALENCE (MAPPRTFORMATOPT (  4:  4), MAPPRINTFORM)
      EQUIVALENCE (FHSMAPOPTIONS   ( 47: 48), MAPLIBLOADOPT)
      EQUIVALENCE (MAPLIBLOADOPT   (  1:  1), MAPLIBLOADMODE)
      EQUIVALENCE (MAPLIBLOADOPT   (  2:  2), MAPLIBLOADFILE)
      EQUIVALENCE (FHSMAPOPTIONS   ( 49: 49), MAPHOLECOLOR)
      EQUIVALENCE (FHSMAPPAR       (221:328), FHSEXITPAR)
      EQUIVALENCE (FHSEXITPAR      (  1:  4), EXITIDENTLEN)
      EQUIVALENCE (FHSEXITPAR      (  5: 12), EXITIDENT)
      EQUIVALENCE (FHSEXITPAR      ( 13: 13), EXITINOUT)
      EQUIVALENCE (FHSEXITPAR      ( 14: 14), EXITRETINFO)
      EQUIVALENCE (FHSEXITPAR      ( 21: 24), EXITFLDLEN)
      EQUIVALENCE (FHSEXITPAR      ( 25: 28), EXITEFFLEN)
      EQUIVALENCE (FHSEXITPAR      ( 29:108), EXITDATA)
*
****************************************************************
```

The individual data fields have the same meaning as in COBOL; see the section “The FHS-MAIN-PAR data structure” on page 303ff.

8.2.2 The FHSINITPAR data structure

FHSINITPAR is the name of the following data structure, which is copied into the program by means of "**%INCLUDE FFOINITP**".

```
************************************************************************
* NAME              FFOINITP
* LANGUAGE          FOR
* VERSION           811
*
*        DATA STRUCTURE FOR THE FHSINIT CALL
*
************************************************************************
*
*
*                                              FHS INIT PAR
      CHARACTER*156 FHSINITPAR
*
*                                                FHS INIT PAR GENERAL
      CHARACTER*16  FHSINITPARGEN
*                                                  FHS I O AREA LEN
      INTEGER*4     FHSIOAREALEN
*                                                  FHS RES MAP NO
      INTEGER*2     FHSRESIMPNO
*                                                  FHS MAP NO
      INTEGER*2     FHSIMPNO
*                                                  FHS ACCESS METHOD
      CHARACTER*1   FHSACCESSMETHOD
*
*                                                FHS MAPPING DEFAULTS
      CHARACTER*60  FHSIMPDEFAULTS
*
*                                                FHS MAP OPTIONS
      CHARACTER*60  FHSIMPOPTIONS
*                                                  MAP DEVICE TYPE
      CHARACTER*4   IMPDEVICETYPE
*                                                  MAP CONTROL UNIT
      CHARACTER*4   IMPCONTROLUNIT
*                                                  MAP USER AREA LEN
      INTEGER*4     IMPUSERAREALEN
*                                                  MAP AUTO TAB
      CHARACTER*1   IMPAUTOTAB
*                                                  MAP EFF LEN
      CHARACTER*1   IMPEFFLEN
*                                                  MAP POS DET CHAR
```

```
      CHARACTER*1   IMPPOSDETCHAR
*
      CHARACTER*1   IMPNEGDETCHAR
*
      CHARACTER*4   IMPREADMETHOD
*
      CHARACTER*1   IMPSCREENPREMOD
*
      CHARACTER*1   IMPREADNILS
*
      CHARACTER*1   IMPUSEALLATTRS
*
      CHARACTER*1   IMPPRINTEROPT
*
      CHARACTER*1   IMPPRTRETBYTE1
*
      CHARACTER*1   IMPPRTRETBYTE2
*
      CHARACTER*3   IMPHARDCOPY
*
      CHARACTER*1   HARDCOPYOPT
*
      INTEGER*2     CENTRALPRTADDR
*
      CHARACTER*1   IMPAUTOHARDCOPY
*
      CHARACTER*1   IMPLOCKKEYS
*
      CHARACTER*1   IMPCLEAROPTION
*
      CHARACTER*1   IMPBELOPTION
*
      CHARACTER*4   IMPPRTFORMOPT
*
      CHARACTER*1   IMPPRINTLINES
*
      CHARACTER*1   IMPPRINTCOLUMNS
*
      CHARACTER*1   IMPPRINTPAPER
*
      CHARACTER*1   IMP PRINT FORM
*
      CHARACTER*2   IMPLIBLOADOPT
*
      CHARACTER*1   IMPLIBLOADMODE
*
      CHARACTER*1   IMPLIBLOADFILE
*
```

```
MAP NEG DET CHAR

MAP READ METHOD

MAP SCREEN PRE MOD

MAP READ NILS

MAP USE ALL ATTRS

MAP PRINTER OPTION

MAP PRINTER RETURN BYTE1

MAP PRINTER RETURN BYTE2

MAP HARDCOPY

  HARDCOPY OPTION

  CENTRAL PRINT ADDR

MAP AUTO HARDCOPY

MAP LOCK KEYS

MAP CLEAR OPTION

MAP BEL OPTION

MAP PRINT FORMAT OPTION

  MAP PRINT LINES

  MAP PRINT COLUMNS

  MAP PRINT PAPER

  MAP PRINT FORM

MAP LIB LOAD OPTION

  MAP LIB LOAD MODE

  MAP LIB LOAD FILE

  MAP HOLE COLOR
```

```
      CHARACTER*1   IMPHOLECOLOR
*
*
*                                                FHS INIT SYS INFO
      CHARACTER*80  FHSINITSYSINFO
*
*
*                                                  FHS BS2000 INFO
      CHARACTER*80  FHSBS2000INFO
*                                                    FHS MAP LIB OPT
      CHARACTER*1   FHSIMPLIBOPT
*                                                    FHS MAP LIB NAME
      CHARACTER*54  FHSIMPLIBNAME
*
*****************************************************************
*
*
      EQUIVALENCE (FHSINITPAR      (  1: 16), FHSINITPARGEN)
      EQUIVALENCE (FHSINITPARGEN   (  1:  4), FHSIOAREALEN)
      EQUIVALENCE (FHSINITPARGEN   (  5:  6), FHSRESIMPNO)
      EQUIVALENCE (FHSINITPARGEN   (  7:  8), FHSIMPNO)
      EQUIVALENCE (FHSINITPARGEN   ( 16: 16), FHSACCESSMETHOD)
      EQUIVALENCE (FHSINITPAR      ( 17: 76), FHSIMPDEFAULTS)
      EQUIVALENCE (FHSIMPDEFAULTS  (  1: 60), FHSIMPOPTIONS)
      EQUIVALENCE (FHSIMPOPTIONS   (  1:  4), IMPDEVICETYPE)
      EQUIVALENCE (FHSIMPOPTIONS   (  5:  8), IMPCONTROLUNIT)
      EQUIVALENCE (FHSIMPOPTIONS   (  9: 12), IMPUSERAREALEN)
      EQUIVALENCE (FHSIMPOPTIONS   ( 13: 13), IMPAUTOTAB)
      EQUIVALENCE (FHSIMPOPTIONS   ( 14: 14), IMPEFFLEN)
      EQUIVALENCE (FHSIMPOPTIONS   ( 15: 15), IMPPOSDETCHAR)
      EQUIVALENCE (FHSIMPOPTIONS   ( 16: 16), IMPNEGDETCHAR)
      EQUIVALENCE (FHSIMPOPTIONS   ( 25: 28), IMPREADMETHOD)
      EQUIVALENCE (FHSIMPOPTIONS   ( 29: 29), IMPSCREENPREMOD)
      EQUIVALENCE (FHSIMPOPTIONS   ( 30: 30), IMPREADNILS)
      EQUIVALENCE (FHSIMPOPTIONS   ( 31: 31), IMPUSEALLATTRS)
      EQUIVALENCE (FHSIMPOPTIONS   ( 32: 32), IMPPRINTEROPT)
      EQUIVALENCE (FHSIMPOPTIONS   ( 33: 33), IMPPRTRETBYTE1)
      EQUIVALENCE (FHSIMPOPTIONS   ( 34: 34), IMPPRTRETBYTE2)
      EQUIVALENCE (FHSIMPOPTIONS   ( 36: 38), IMPHARDCOPY)
      EQUIVALENCE (IMPHARDCOPY     (  1:  1), HARDCOPYOPT)
      EQUIVALENCE (IMPHARDCOPY     (  2:  3), CENTRALPRTADDR)
      EQUIVALENCE (FHSIMPOPTIONS   ( 39: 39), IMPAUTOHARDCOPY)
      EQUIVALENCE (FHSIMPOPTIONS   ( 40: 40), IMPLOCKKEYS)
      EQUIVALENCE (FHSIMPOPTIONS   ( 41: 41), IMPCLEAROPTION)
      EQUIVALENCE (FHSIMPOPTIONS   ( 42: 42), IMPBELOPTION)
      EQUIVALENCE (FHSIMPOPTIONS   ( 43: 46), IMPPRTFORMOPT)
      EQUIVALENCE (IMPPRTFORMOPT   (  1:  1), IMPPRINTLINES)
      EQUIVALENCE (IMPPRTFORMOPT   (  2:  2), IMPPRINTCOLUMNS)
```

```
      EQUIVALENCE (IMPPRTFORMOPT   (  3:  3), IMPPRINTPAPER)
      EQUIVALENCE (IMPPRTFORMOPT   (  4:  4), IMPPRINTFORM)
      EQUIVALENCE (FHSIMPOPTIONS   ( 47: 48), IMPLIBLOADOPT)
      EQUIVALENCE (IMPLIBLOADOPT   (  1:  1), IMPLIBLOADMODE)
      EQUIVALENCE (IMPLIBLOADOPT   (  2:  2), IMPLIBLOADFILE)
      EQUIVALENCE (FHSIMPOPTIONS   ( 49: 49), IMPHOLECOLOR)
      EQUIVALENCE (FHSINITPAR      ( 77:156), FHSINITSYSINFO)
      EQUIVALENCE (FHSINITSYSINFO  (  1: 80), FHSBS2000INFO)
      EQUIVALENCE (FHSBS2000INFO   ( 26: 26), FHSIMPLIBOPT)
      EQUIVALENCE (FHSBS2000INFO   ( 27: 80), FHSIMPLIBNAME)
*
*****************************************************************
```

The individual data fields have the same meaning as in COBOL; see the section “The FHS-INIT-PAR data structure” on page 330ff.

8.2.3 The FHSATTRPAR data structure

FHSATTRPAR is the name of the following data structure, which is copied into the program by means of "**%INCLUDE FFOATTRP**".

```
*****************************************************************
*     FHSATTRP   V811    FOR1                                     *
*       DATA STRUCTURE FOR THE FHSATTR CALL                      *
*                                                                *
*****************************************************************
*
*                                        FHS ATTR PAR
      CHARACTER*40  FHSATTRPAR
*                                        FHS ATTR PAR BASIC
      CHARACTER*16  FHSATTRPARBASIC
*                                        A UPDATE METHOD
      CHARACTER*3   AUPDATEMETHOD   / 'REP' /
*                                        A PROT LEVEL
      CHARACTER*4   APROTLEVEL
*                                        A DISP LEVEL
      CHARACTER*1   ADISPLEVEL
*                                        FHS ATTR PAR OPTIONAL
      CHARACTER*24  FHSATTRPAROPTI
*                                        FHS ATTR PAR OPTIONS
      CHARACTER*24  FHSATTRPAROPT
*                                        A NO HARDCOPY
      CHARACTER*1   ANOHARDCOPY
*                                        A NUMERIC
      CHARACTER*1   ANUMERIC
*                                        A SIGNAL
      CHARACTER*1   ASIGNAL
*                                        A ITALIC
      CHARACTER*1   AITALIC
*                                        A WIDE
      CHARACTER*1   AWIDE
*                                        A TALL
      CHARACTER*1   ATALL
*                                        A SKIP
      CHARACTER*1   ASKIP
*****************************************************************
      EQUIVALENCE (FHSATTRPAR      (  1: 16), FHSATTRPARBASIC)
      EQUIVALENCE (FHSATTRPARBASIC (  1:  3), AUPDATEMETHOD)
      EQUIVALENCE (FHSATTRPARBASIC (  9: 12), APROTLEVEL)
      EQUIVALENCE (FHSATTRPARBASIC ( 13: 13), ADISPLEVEL)
      EQUIVALENCE (FHSATTRPAR      ( 17: 40), FHSATTRPAROPTI)
      EQUIVALENCE (FHSATTRPAROPTI  (  1: 24), FHSATTRPAROPT)
      EQUIVALENCE (FHSATTRPAROPT   (  1:  1), ANOHARDCOPY)
```

```
      EQUIVALENCE (FHSATTRPAROPT   (  2:  2), ANUMERIC)
      EQUIVALENCE (FHSATTRPAROPT   (  3:  3), ASIGNAL)
      EQUIVALENCE (FHSATTRPAROPT   (  4:  4), AITALIC)
      EQUIVALENCE (FHSATTRPAROPT   ( 17: 17), AWIDE)
      EQUIVALENCE (FHSATTRPAROPT   ( 18: 18), ATALL)
      EQUIVALENCE (FHSATTRPAROPT   ( 24: 24), AASKIP)
*
****************************************************************
```

The individual data fields have the same meaning as in COBOL; see the section “The FHS-ATTR-PAR data structure” on page 334ff.

8.2.4 The FHSEXITMODPAR data structure

FHSEXITMODPAR is the name of the following data structure, which is copied into the exit routine by means of "**%INCLUDE FFOEXITP,'EXIT'='EXITF'**".

```
************************************************************************
* NAME              FFOEXITP
* LANGUAGE          FOR
* VERSION           811
*
*        DATA STRUCTURE FOR THE EXIT ROUTINE
*
************************************************************************
*
*                                                  FHS EXITMOD PAR
      CHARACTER*108 FHSEXITMODPAR
*                                                    EXITMOD PAR
      CHARACTER*108 EXITMODPAR
*                                                      FHS EXIT PAR
      CHARACTER*108 FHSEXITPAR
*                                                        EXIT IDENT LEN
      INTEGER*4     EXITIDENTLEN
*                                                        EXIT IDENT
      CHARACTER*8   EXITIDENT
*                                                        EXIT IN OUT
      CHARACTER*1   EXITINOUT
*                                                        EXIT RET INFO
      CHARACTER*1   EXITRETINFO
*                                                        EXIT FLD LEN
      INTEGER*4     EXITFLDLEN
*                                                        EXIT EFF LEN
      INTEGER*4     EXITEFFLEN
*                                                        EXIT DATA
      CHARACTER*80  EXITDATA
*
*****************************************************************
*
      EQUIVALENCE (FHSEXITMODPAR   (  1:108), EXITMODPAR)
      EQUIVALENCE (EXITMODPAR      (  1:108), FHSEXITPAR)
      EQUIVALENCE (FHSEXITPAR      (  1:  4), EXITIDENTLEN)
      EQUIVALENCE (FHSEXITPAR      (  5: 12), EXITIDENT)
      EQUIVALENCE (FHSEXITPAR      ( 13: 13), EXITINOUT)
      EQUIVALENCE (FHSEXITPAR      ( 14: 14), EXITRETINFO)
      EQUIVALENCE (FHSEXITPAR      ( 21: 24), EXITFLDLEN)
      EQUIVALENCE (FHSEXITPAR      ( 25: 28), EXITEFFLEN)
      EQUIVALENCE (FHSEXITPAR      ( 29:108), EXITDATA)
*
*****************************************************************
```

The individual data fields have the same meaning as in COBOL; see the section “The FHS-EXITMOD-PAR data structure” on page 338ff.

8.2.5 The FHSCCSNPAR data structure

FHSCCSNPAR is the name of the following data structure, which is copied into the application program by means of "**%INCLUDE FFOCCSNP**". The FHSSERV call needs this data structure to determine the character set name of a format.

```
************************************************************************
*     FHSCCSNP   V811    FOR1                                          *
*        DATA STRUCTURE FOR THE FHSSERV CALL                           *
*                                                                      *
************************************************************************
*
*
*                                          FHS CCSN PAR
      CHARACTER*32  FHSCCSNPAR
*
      CHARACTER*8   FHSCCSNINFO
*
************************************************************************
*
      EQUIVALENCE (FHSCCSNPAR      (  9: 16), FHSCCSNINFO)
*
************************************************************************
```

FHS enters the name of the character set in the FHSCCSNINFO field.

8.3 Fortran calls for TIAM

When using FHS, the TIAM calls WROUT and WRTRD can be issued in Fortran programs.

8.3.1 TIAM call WROUT

The TIAM call “CALL WROUT” is used to output formatted messages.

```
CALL WROUT(TIAMCONTROLINFO,transfer-area,FHSMAINPAR)
```

The parameters have the following meaning:

TIAMCONTROLINFO

Fortran data structure which controls the TIAM call. This data structure is shown in the “TIAM (TRANSDATA, BS2000)” User Guide.

transfer-area

Name of the data transfer area. The first field of this area must be a variable in which FHS writes the length of the message. This can be done as follows:

```
CHARACTER * nn FORMAT$EA
INTEGER * 4 DTLNGT
%INCLUDE FORMAT
EQUIVALENCE (FORMAT$EA (1:4), DTLNGT)
EQUIVALENCE (FORMAT$EA (5:nn), FORMAT)
```

FORMAT$EA is the name of the data transfer area and nn is the length of the format + 4. The length of the format is contained in the first line (comment line) of the addressing aid (include element) generated by IFG.

FHSMAINPAR

controls formatting. Please refer to the section “TIAM calls” on page 343 for a description of the entries you have to make in the individual fields and the return codes supplied by FHS.

8.3.2 TIAM call WRTRD

The TIAM call “CALL WRTRD” is used to input/output formatted messages.

```
CALL WRTRD(TIAMCONTROLINFO,transfer-area1,transfer-area2,FHSMAINPAR)
```

The parameters have the following meaning:

TIAMCONTROLINFO

Fortran data structure which controls the TIAM call. This data structure is described in the “TIAM (TRANSDATA, BS2000)” User Guide.

transfer-area

Name of the data transfer area for output and input. The same area must be specified for both because Fortran can only use #formats. The first field of this area must be a variable, in which FHS writes the length of the message. This can be done as follows:

```
CHARACTER * nn FORMAT$EA
INTEGER * 4 DTLNGT
%INCLUDE FORMAT
EQUIVALENCE (FORMAT$EA (1:4), DTLNGT)
EQUIVALENCE (FORMAT$EA (5:nn), FORMAT)
```

FORMAT$EA is the name of the data transfer area and nn is the length of the format + 4. The length of the format is contained in the first line (comment line) of the addressing aid (include element) generated by IFG.

FHSMAINPAR

controls formatting. Please refer to the section “TIAM calls” on page 343 for a description of the entries you have to make in the individual fields and the return codes supplied by FHS.

8.4 FHS Fortran calls

With Fortran you can use the subprogram calls FHSCURS (position cursor), FHSATTR (update +format attribute), FHSINIT (initialize formatting), and FHSSERV (initialize data transfer area and determine name of character set). o

8.4.1 FHSCURS

For *formats and +formats, you can position the cursor in any unprotected ooor selectable field of the format by means of the FHSCURS CALL. FHSCURS is called as follows:

```
CALL  FHSCURS(FHSMAINPAR,fieldname)
```

“fieldname” is the name of the field in which FHS is to position the cursor. FHSMAINPAR contains the return codes of the FHSCURS call. For further information please refer to the section “CALL “FHSCURS”” on page 361.

8.4.2 FHSATTR

For +formats you can update the attribute fields of a field and thus the field attributes by means of FSHATTR. FHSATTR is called as follows:

```
CALL  FHSATTR(FHSCONTROLINFO,FHSATTRPAR,attributefield);
```

“attributefield” is the name of the attribute field you wish to modify. FHSCONTROLINFO is the part of the FHSMAINPAR data structure that contains the return code; FHSATTRPAR controls the modification of attributes. For further information see the section “CALL “FHSATTR”” on page 363.

8.4.3 FHSINIT

The FHSINIT subprogram serves to initialize formatting and to specify the formats to be loaded upon opening formatting. The start parameters for #formats are defined with this call. CALL FHSINIT is necessary

- if you wish to work with formats that are to be loaded upon opening, or
- if you wish to use more than 100 different formats.

Furthermore, the FHSINIT call can be used to define your own formatting standard. This standard can be modified for every formatting operation.

If CALL FHSINIT is issued more than once, the data area FHS_INIT_PAR_GENERAL is not evaluated for any but the first call, since formatting is initiated by the first CALL “FHSINIT”. FHSINIT is called as follows:

```
CALL FHSINIT(FHSCONTROLINFO,FHSINITPAR,area,[format-list]);
```

FHSCONTROLINFO is part of the FHSMAINPAR data structure (see page 425), FHSINITPAR is shown on page 431. “area” may be any area; it must be specified for compatibility reasons. “format-list” is the name of a list containing the names of the formats to be loaded upon opening formatting.

Please refer to the section “CALL “FHSINIT”” on page 366, for the entries to be made in the individual fields. Except for the underscores, the field names are identical with the COBOL field names.

8.4.4 FHSSERV

The FHSSERV subprogram enables you to use four FHS service functions:

– ’Initialization of the Data Transfer Area’ for #formats
– ’Determine Name of Character Set’
– ‘Unload Format’
– ‘Dynamic Retrieval of Information on the Structure of the Addressing Aid for #Formats’

8.4.4.1 Initialization of the data transfer area

All field attributes are set according to their default values in the format. Neither the global attributes (except for ‘Formatting acknowledgments’) nor the field contents are updated. It is thus possible to reset to the initial status data transfer areas that have already been supplied with data. FHSSERV is called as follows to initialize the data transfer area:

```
CALL FHSSERV(FHSMAINPAR,transfer-area);
```

The FHSMAINPAR data structure is shown on page 425. “transfer-area” is the name of the transfer area without the preceding length field.

Please refer to the section “CALL “FHSSERV”” on page 370 for the entries to be made in the individual fields. Except for the hyphens, the field names are identical with the COBOL field names.

8.4.4.2 Determine name of character set

The format specified in FHSMAINPAR is loaded and the name of the corresponding character set is entered in the field FHSCCSNINFO of the data structure FHSCCSNPAR. FHSSERV is called as follows for this function:

```
CALL FHSSERV(FHSMAINPAR,FHSCCSNPAR)
```

The FHSMAINPAR data structure is described on page 425; FHSCCSNPAR is described on page 439.

8.4.4.3 Unload format

The format specified in FHSMAINPAR is unloaded and can be replaced by a modified format.

FHSSERV is called as follows for this function:

```
CALL FHSSERV(FHSMAINPAR,transfer-area)
```

The FHSMAINPAR data structure is described on page 425. “transfer-area” is the name of the transfer-area without the preceding length field.

Please refer to the section “CALL “FHSSERV”” on page 370 for the entries to be made in the individual fields. Except for the hyphens, the field names are identical with the COBOL field names.

8.4.4.4 Dynamic retrieval of information on the structure of the addressing aid for #formats

The structure of the addressing aid for the format specified in FHSMAINPAR is returned in the transfer area.

FHSSERV is called as follows for this function:

```
CALL FHSSERV(FHSMAINPAR,transfer-area)
```

The FHSMAINPAR data structure is described on page 425. “transfer-area” is the name of the transfer-area without the preceding length field.

Please refer to page 375 for the entries to be made in the individual fields. Except for the hyphens, the field names are identical with the COBOL field names.

8.5 Attribute modification

8.5.1 List of attribute values FHSATTRIBUTEVALUES

The name of the following list of attribute values is FHSATTRIBUTEVALUES; this is copied into the application program by means of "**%INCLUDE FFOAVAL**". It generates symbolically addressable attribute values for the global attributes and field attributes of the data transfer area with separate attribute blocks and field contents.

```
*****************************************************************
*     FHSAVAL  V811   FOR1                                      *
*                                                               *
*     ATTRIBUTE VALUES FOR IFG ADDRESSING AID                   *
*                                                               *
*****************************************************************
*     FHS-ATTRIBUTE-VALUES                                      *
*****************************************************************
*
*     GLOBAL ATTRIBUTE VALUES (CHARACTERS)
*     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*                                              GA-DEFAULT-VALUES
      CHARACTER*1  GADEFAULT         / ' ' /
*                                              FORMATTING-INDICATORS
*                                                FIELDS-MODIFICATION
      CHARACTER*1  GAMODIFIED        / 'Y' /
      CHARACTER*1  GANOTMODIFIED     / ' ' /
*                                                FIELDS-DETECTION
      CHARACTER*1  GADETECTED        / 'Y' /
      CHARACTER*1  GANOTDETECTED     / ' ' /
*                                                FIELDS-VALIDATION
      CHARACTER*1  GAVALID           / 'V' /
      CHARACTER*1  GANOTVALID        / ' ' /
*                                                FIELDS-UNDEFINED
      CHARACTER*1  GAUNDEFINED       / 'Y' /
      CHARACTER*1  GANOTUNDEFINED    / ' ' /
*                                              INPUT-IDENTIFICATION
*                                                INPUT-KEY-CLASS
      CHARACTER*1  GAINPUTKEY        / 'I' /
      CHARACTER*1  GAFKEY            / 'F' /
      CHARACTER*1  GAKKEY            / 'K' /
      CHARACTER*1  GAPOSRM           / 'P' /
      CHARACTER*1  GANEGRM           / 'N' /
      CHARACTER*1  GAINPUTNONE       / ' ' /
```

```
*                                                  DEVICE-CONTROLS
*                                                    INIT-CONTROL
      CHARACTER*1  GANOINIT          / 'N' /
      CHARACTER*1  GAFIRSTINIT       / 'F' /
      CHARACTER*1  GALASTINIT        / 'L' /
      CHARACTER*1  GABOTHINIT        / 'B' /
*                                                    TABULATOR-CONTROL
      CHARACTER*1  GAAUTOTAB         / 'A' /
      CHARACTER*1  GANOAUTOTAB       / 'N' /
*                                                    FUNCTION-LOCK
      CHARACTER*1  GAKEYLOCK         / 'K' /
*                                                    VMI-CONTROL
      CHARACTER*1  GAVMI1            / '1' /
      CHARACTER*1  GAVMI2            / '2' /
      CHARACTER*1  GAVMI3            / '3' /
*                                                    HMI-CONTROL
      CHARACTER*1  GAHMI1            / '1' /
      CHARACTER*1  GAHMI2            / '2' /
      CHARACTER*1  GAHMI3            / '3' /
*                                                  OUTPUT-CONTROLS
*                                                    CYCLE-CONTROL
      CHARACTER*1  GACLOSE           / 'C' /
*                                                    COPY-CONTROL
      CHARACTER*1  GAHARDCOPYGEN     / 'H' /
      CHARACTER*1  GAHARDCOPYLOC     / 'L' /
*                                                    ALARM-CONTROL
      CHARACTER*1  GAALARM           / 'A' /
*                                                    HOLE-COLOR
      CHARACTER*1  GANOCOLOR         / 'U' /
      CHARACTER*1  GAGREYHOLE        / 'G' /
      CHARACTER*1  GAWHITEHOLE       / 'W' /
*                                                  FORMATTING-CONTROLS
*                                                    DISPLAY-SELECTION
      CHARACTER*1  GABOXB            / 'B' /
      CHARACTER*1  GABOXL            / 'C' /
      CHARACTER*1  GAKEB             / 'K' /
      CHARACTER*1  GAKEL             / 'L' /
*                                                    LEVEL-SELECTION
      CHARACTER*1  GALEVEL1          / '1' /
      CHARACTER*1  GALEVEL2          / '2' /
      CHARACTER*1  GALEVEL3          / '3' /
      CHARACTER*1  GALEVELP          / 'P' /
*                                                    OUTPUT-MODE
      CHARACTER*1  GARDIF            / 'R' /
*                                                    CURSOR-CONTROL
      CHARACTER*1  GAFIELDCURSOR     / 'F' /
      CHARACTER*1  GAEDITCURSOR      / 'E' /
      CHARACTER*1  GARELCURSOR       / 'R' /
```

```
*                                                    USER-EXIT-CONTROL
      CHARACTER*1  GANOUEXIT          / 'N' /
      CHARACTER*1  GAOUTUEXIT         / 'O' /
      CHARACTER*1  GAINUEXIT          / 'I' /
      CHARACTER*1  GABOTHUEXIT        / 'B' /
*
*     FIELD ATTRIBUTE VALUES (CHARACTERS)
*     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*                                                 DEFAULT-VALUES
      CHARACTER*1  FADEFAULT          / ' ' /
*
*                                                 BASIC-ATTRIBUTES
*                                                   INPUT-STATE-/
*                                                   -INPUT-STATE-ACT
      CHARACTER*1  FAMODIFIED         / 'M' /
      CHARACTER*1  FACLEARED          / 'C' /
      CHARACTER*1  FADETECTED         / 'D' /
      CHARACTER*1  FAUNDEFINED        / 'U' /
      CHARACTER*1  FANOTTOUCHED       / ' ' /
*                                                   EDIT-STATE
      CHARACTER*1  FAVALID            / 'V' /
      CHARACTER*1  FAINVALID          / 'I' /
      CHARACTER*1  FAMUSTERROR        / 'M' /
      CHARACTER*1  FANOTCHECKED       / ' ' /
*                                                   OUTPUT-CONTROL
      CHARACTER*1  FAOUTPUTINIT       / 'I' /
      CHARACTER*1  FAOUTPUTDATA       / 'D' /
      CHARACTER*1  FAOUTPUTUNDEF      / 'U' /
*
*                                                 FIELD-INPUT
*                                                   INPUT-CONTROL
      CHARACTER*1  FANORMALIN         / 'N' /
      CHARACTER*1  FAMUSTIN           / 'M' /
      CHARACTER*1  FAPOTMUSTIN        / 'P' /
      CHARACTER*1  FAAUTORETIN        / 'A' /
*                                                   PROTECTION
      CHARACTER*1  FAUNPROTECTED      / 'U' /
      CHARACTER*1  FAPROTECTED        / 'P' /
      CHARACTER*1  FAASKIP            / 'A' /
      CHARACTER*1  FADETECTABLE       / 'D' /
*
*                                                 DISPLAY-CONTROL
*                                                   INTENSITY
      CHARACTER*1  FAHIGHINTENSITY    / 'H' /
      CHARACTER*1  FANORMALINTENS     / 'N' /
```

```
*                                                    VISIBILITY
      CHARACTER*1  FAVISIBLE         / 'V' /
      CHARACTER*1  FASIGNALING       / 'S' /
      CHARACTER*1  FAINVISIBLE       / 'I' /
*                                                    UNDERLINE
      CHARACTER*1  FAUNDERLINED      / 'Y' /
      CHARACTER*1  FANOTUNDERLINED   / 'N' /
*                                                    INVERSE
      CHARACTER*1  FAINVERSE         / 'Y' /
      CHARACTER*1  FANOTINVERSE      / 'N' /
*
*                                                 COLOUR
      CHARACTER*1  FARED             / '1' /
      CHARACTER*1  FAGREEN           / '2' /
      CHARACTER*1  FAYELLOW          / '3' /
      CHARACTER*1  FABLUE            / '4' /
      CHARACTER*1  FAMAGENTA         / '5' /
      CHARACTER*1  FACYAN            / '6' /
      CHARACTER*1  FAWHITE           / '7' /
      CHARACTER*1  FANOCOLOUR        / 'N' /
*
*                                                 CURSOR
      CHARACTER*1  FACURSOR          / 'Y' /
      CHARACTER*1  FAHOLDCURSOR      / 'H' /
      CHARACTER*1  FANOCURSOR        / 'N' /
*****************************************************************
```

8.5.2 The FHSATTRIBUTEMOVE data structure

This section describes the attribute modifications for +formats. For #formats, this section is only relevant if the field attribute group 'attribute combination' is used (see page 71ff).

FHS offers the Fortran programmer two possibilities for modifying the field attributes of the format fields in +formats in the program:

– by means of "FHSATTR" (see page 442ff) or

– by means of the COPY element **FHSATTRIBUTEMOVE**. If a suitable attribute combination exist in this data structure, you can enter it in the desired attribute field by means of a simple statement.

FHSATTRIBUTEMOVE is the name of the following data structure, which is copied into the application program by means of "**%INCLUDE FFOATTRM**".

```
************************************************************
*                                                          *
*    FHSATTRM Version 811                                  *
*                                                          *
*       DATA STRUCTURE FOR THE ATTRIBUTE MOVE              *
*                                                          *
************************************************************
** KCALPH   ** UNPROT,BRT,PRINT
      INTEGER*2     KCALPH            /20512/
** KCNUME   ** UNPROT,BRT,NUM
      INTEGER*2     KCNUME            /21024/
** KCPROT   ** PROT,NORM
      INTEGER*2     KCPROT            /4360/
** KCUNPR   ** UNPROT,BRT
      INTEGER*2     KCUNPR            /20512/
** KCNINT   ** UNPROT,NORM
      INTEGER*2     KCNINT            /20488/
** KCDINT   ** UNPROT,DRK
      INTEGER*2     KCDINT            /20484/
** KCHINT   ** UNPROT,BRT
      INTEGER*2     KCHINT            /20512/
** KCITAL   ** UNPROT,BRT,ITAL
      INTEGER*2     KCITAL            /20514/
** KCSIGN   ** UNPROT,BRT,SIGN
      INTEGER*2     KCSIGN            /20513/
** KCDETE   ** PROT,BRT,DET
      INTEGER*2     KCDETE            /6432/
** KCPREM   ** FSET,BRT
      INTEGER*2     KCPREM            /5152/
** KCAUN    ** UNPROT,NORM
      INTEGER*2     KCAUN             /20488/
```

```
** KCNUN    ** UNPROT,NORM,NUM
      INTEGER*2     KCNUN             /21000/
** KCAPN    ** PROT,NORM
      INTEGER*2     KCAPN             /4360/
** KCNPN    ** PROT,NORM,NUM
      INTEGER*2     KCNPN             /4872/
** KCAUD    ** UNPROT,DRK
      INTEGER*2     KCAUD             /20484/
** KCNUD    ** UNPROT,DRK,NUM
      INTEGER*2     KCNUD             /20996/
** KCAPD    ** PROT,DRK
      INTEGER*2     KCAPD             /4356/
** KCNPD    ** PROT,DRK,NUM
      INTEGER*2     KCNPD             /4868/
** KCAUH    ** UNPROT,BRT
      INTEGER*2     KCAUH             /20512/
** KCNUH    ** UNPROT,BRT,NUM
      INTEGER*2     KCNUH             /21024/
** KCAPH    ** PROT,BRT
      INTEGER*2     KCAPH             /4384/
** KCNPH    ** PROT,BRT,NUM
      INTEGER*2     KCNPH             /4896/
** KCAUI    ** UNPROT,BRT,ITAL
      INTEGER*2     KCAUI             /20514/
** KCNUI    ** UNPROT,BRT,ITAL,NUM
      INTEGER*2     KCNUI             /21026/
** KCAPI    ** PROT,NORM,ITAL
      INTEGER*2     KCAPI             /4362/
** KCNPI    ** PROT,NORM,ITAL,NUM
      INTEGER*2     KCNPI             /4874/
** KCAUS    ** UNPROT,BRT,SIGN
      INTEGER*2     KCAUS             /20513/
** KCNUS    ** UNPROT,BRT,SIGN,NUM
      INTEGER*2     KCNUS             /21025/
** KCAPS    ** PROT,NORM,SIGN
      INTEGER*2     KCAPS             /4361/
** KCNPS    ** PROT,NORM,SIGN,NUM
      INTEGER*2     KCNPS             /4873/
** KCAUND   ** UNPROT,NORM,DET
      INTEGER*2     KCAUND            /22536/
** KCNUND   ** UNPROT,NORM,DET
      INTEGER*2     KCNUND            /22536/
** KCAPND   ** PROT,NORM,DET
      INTEGER*2     KCAPND            /6408/
** KCNPND   ** PROT,NORM,DET,NUM
      INTEGER*2     KCNPND            /6920/
** KCAUHD   ** UNPROT,BRT,DET
      INTEGER*2     KCAUHD            /22560/
```

```
** KCNUHD   ** UNPROT,BRT,DET
      INTEGER*2     KCNUHD            /22560/
** KCAPHD   ** PROT,BRT,DET
      INTEGER*2     KCAPHD            /6432/
** KCNPHD   ** PROT,BRT,DET,NUM
      INTEGER*2     KCNPHD            /6944/
** KCAUID   ** UNPROT,BRT,DET,ITAL
      INTEGER*2     KCAUID            /22562/
** KCNUID   ** UNPROT,BRT,DET,ITAL
      INTEGER*2     KCNUID            /22562/
** KCAPID   ** PROT,NORM,DET,ITAL
      INTEGER*2     KCAPID            /6410/
** KCNPID   ** PROT,NORM,DET,ITAL,NUM
      INTEGER*2     KCNPID            /6922/
** KCAUSD   ** UNPROT,BRT,DET
      INTEGER*2     KCAUSD            /22560/
** KCNUSD   ** UNPROT,BRT,DET
      INTEGER*2     KCNUSD            /22560/
** KCAPSD   ** PROT,NORM,DET
      INTEGER*2     KCAPSD            /6408/
** KCNPSD   ** PROT,NORM,DET,NUM
      INTEGER*2     KCNPSD            /6920/
** KCAUNP   ** FSET,NORM
      INTEGER*2     KCNAUNP           /5128/
** KCNUNP   ** FSET,NORM,NUM
      INTEGER*2     KCNUNP            /5640/
** KCAPNP   ** PROTRET,NORM
      INTEGER*2     KCAPNP            /12296/
** KCNPNP   ** PROTRET,NORM,NUM
      INTEGER*2     KCNPNP            /12808/
** KCAUHP   ** FSET,BRT
      INTEGER*2     KCAUHP            /5152/
** KCNUHP   ** FSET,BRT,NUM
      INTEGER*2     KCNUHP            /5664/
** KCAPHP   ** PROTRET,BRT
      INTEGER*2     KCAPHP            /12320/
** KCNPHP   ** PROTRET,BRT,NUM
      INTEGER*2     KCNPHP            /12832/
** KCPBSP   ** PROT,BRT,SIGN
      INTEGER*2     KCPBSP            /4385/
*****************************************************************
```

8.6 Compiler-dependent constraints

FHS Fortran programs can only be generated in XS format with TIAM version V11.0 or later. The following compiler-dependent constraints therefore apply :

– The parameter “XS=NO” must be specified in the COMOPT statement when compiling the Fortran program with the Fortran compiler FOR1 V2.1A.

– If you wish to compile the Fortran program with the Fortran compiler FOR1 V2.2, you must use TIAM V11.0 because FOR1 always generates XS format.

8.7 Sample program with FHS Fortran

The sample program displays the format shown below on the screen and reads the subsequently made entries. The format has two input fields and one output field.

```
SCREEN FOR TEST Fortran INCLUDE MEMBERS FHS V80

INPUT FIELD 1 : @@@@@@@@@@
INPUT FIELD 2 : @@@@@@@@@@
OUTPUT FIELD  : @@@@@@@@@@
```

The example is implemented by the following Fortran program:

```
      PROGRAM PGMFOR
*
      CHARACTER * 98 DTA
      INTEGER * 4 DTLNGT
      %INCLUDE PGM.LIB(FOR)
      EQUIVALENCE (DTA (1:4)   ,DTLNGT)
      EQUIVALENCE (DTA (5:56)  ,FORGLOBALS)
      EQUIVALENCE (DTA (57:68) ,FORATTR)
      EQUIVALENCE (DTA (69:98) ,FORDATA)

      %INCLUDE FOR.080.(FFOMAINP)
      %INCLUDE FOR.080.(TIFOINFO)
*
      EXTERNAL WRTRD
*
      FHSMAPNAME = 'FOR     '
      FHSRESTARTOPT1 ='Y'
      FHSMAPLIBOPT='Y'
      FHSMAPLIBNAME='PGM.LIB'
      EDITMODEOUT='F'
      OUTPUT='MESSFRPGM '
      WRITE(*,*) 'START OF PROGRAM'
      CALL WRTRD(TIAMCONTROLINFO,DTLNGT,DTLNGT,FHSMAINPAR)
      WRITE(*,*) 'RCCATEGO  :',FORRCCATEGO
      WRITE(*,*) 'RCREASON  :',FORRCREASON
      WRITE(*,*) 'TIAMRC    :',TIAMRC
      WRITE(*,*) 'FHSMAINRC :',FHSMAINRC
      WRITE(*,*) 'ERCATEGO  :',ERRORCATEGORY
      WRITE(*,*) 'ERREASON  :',ERRORREASON
      WRITE(*,*) 'CONTENT OF THE FIELDS :'
      WRITE(*,*) 'FIELD1    :',FIELD1
      WRITE(*,*) 'FIELD2    :',FIELD2
      WRITE(*,*) 'FIELDOUT  :',OUTPUT
      END
```

9 Use of FHS in PL/I programs

This chapter contains all you need to know if you wish to use FHS in PL/I programs with the TIAM access method. The required PL/I data structures are listed as of page 459.

9.1 Structure of FHS PL/I programs

The FHS PL/I interface enables the PL/I programmer to implement FHS to implement FHS functions in TIAM application programs without having to write an ASSEMBLER subprogram for the formatting. The formatting functions have been integrated into the calls of the access method. You merely have to supply certain data structures with data before entering the input/output call. FHS uses these structures to obtain formatting parameters and to store return codes. Formatting is possible using the following TIAM calls (see page 467f):

– CALL “WROUT”
– CALL “WRTRD”

In addition, you can make use of the following FHS PL/I calls, implemented in the form of subprogram calls (CALL...):

– CALL “FHSCURS” for explicitly positioning the cursor in +formats and *formats (see page 469).
– CALL “FHSATTR” for modifying field attributes in +formats (see page 469).
– CALL “FHSINIT” for initiating the formatting and defining certain start parameters for formatting (see page 469)
– CALL “FHSSERV” for calling special FHS service functions (see page 470).

Attributes can be modified with the aid of the include member FP1AVAL which contains a complete list of the symbolic attribute names.

You generate the formats using IFG.

The data structures used by FHS PL/1 are provided in the form of include members. This simplifies the transfer of the formatting parameters to FHS. These data structures are described in the section starting on page 457. Points to be observed when compiling and linking FHS PL/I programs are given in the section starting on page 481.

FHS PL/I programs have the following structure:

```
 PROG1: PROCEDURE;                              program name
/*****************************************************************************/
      %INCLUDE FP1MAINP;                        ready FHS-MAIN-PAR
     [%INCLUDE FP1ATTRP;]                       ready FHS-ATTR-PAR
     [%INCLUDE FP1ATTRM;]                       include member for ATTRIBUT-MOVE
     [%INCLUDE FP1INITP;]                       ready FHS-INIT-PAR
     [%INCLUDE FP1AVAL;]                        ready FHS-ATTRIBUTE-VALUES
     [%INCLUDE FP1EXITP;]                       ready FHS-EXITMOD-PAR
     [%INCLUDE FFOCCSNP;]                       ready FHS-CCSM-PAR
     [%INCLUDE TIP1INFO;]                       TIAM-CONTROL-INFO
/*****************************************************************************/
 DECLARE YYYYYY ENTRY EXTERNAL;                 name of TIAM call
/*****************************************************************************/
     [CALL FHSINIT(...);]                       define start parameters
        .
        .
        .
    [CALL FHSCURS(...);]                      position cursor
        .
        .
    CALL YYYYYY(...);                         TIAM call(s)
        .
    attribute modifications using FP1AVAL}
        or
    CALL FHSATTR(...);
        .

/*****************************************************************************/
```

Optional entries appear in brackets. The include members are stored in the FHS library SYSLIB.FHS... .

9.2 Data structures used by FHS PL/I

Data structures provide the interface between FHS and the application program. They are stored as include elements in the FHS library (SYSLIB.FHS.082) from where they are copied into the application program. In the PL/I program, they are specified in the CALL Macro.

Except for the underscores, the field names of the PL/I data structures are identical with the COBOL field names. You are therefore referred to the relevant section of the COBOL description.

The following data structures are available:

FHS_MAIN_PAR

This data structure is divided into two parts. In “FHS_CONTROL_INFO”, the application program is supplied with information about the formatting run (return codes etc.). “FHS_MAP_PAR” is used by the application program to control formatting.

The FHS_MAIN_PAR data structure is copied into the program using the PL/I statement **%INCLUDE FP1MAINP**. FHS_MAIN_PAR is shown on page 459.

FHS_INIT_PAR

This data structure can be used in the FHSINIT call to supply application program-specific default values for subsequent formatting operations and to specify the format application file.

The FHS_INIT_PAR data structure is copied into the application program by means of the PL/I statement **%INCLUDE FP1INITP**. FHS_INIT_PAR is shown on page 462.

FHS_ATTR_PAR

This data structure is required when using the FHSATTR call for the modification of attributes of +formats.

The FHS_ATTR_PAR data structure is copied into the application program by means of the PL/I statement **%INCLUDE FP1ATTRP**. FHS_ATTR_PAR is shown on page 464.

FHS_EXITMOD_PAR

This data structure corresponds to the user exit interface and is only required when exit routines are used (see the section “Checking data fields with an exit routine” on page 41). FHS_EXITMOD_PAR is copied into the program by means of the PL/I statement **%INCLUDE FP1EXITP**. FHS_EXITMOD_PAR is shown on page 465.

FHS_CCSN_PAR

This data structure is required for the “FHSSERV” call in order to receive the name of the character sets used in the format. It is copied into the application using the PL/I statement **INCLUDE FP 1CCSNP**.

FHS_ATTRIBUTE_VALUES

This list generates symbolically addressable attribute values for the global attributes and field attributes of the data transfer area with separate attribute blocks and field contents. It is copied into the application program by means of the PL/I statement **%INCLUDE FP1AVAL**. FHS_ATTRIBUTE_VALUES is shown on page 472.

FHS_ATTR_MOVE

This data structure is required when modifying the attributes of +formats by assignment (instead of the FHSATTR call).

The FHS_ATTR_MOVE data structure is copied into the application program using the PL/I statement **%INCLUDE FP1ATTRM**. FHS_ATTR_MOVE is shown on page 476.

The table on page 302 shows which data structures are required for which FHS Fortran calls.

9.2.1 The FHS_MAIN_PAR data structure

FHS_MAIN_PAR is the name of the following data structure, which is copied into the program by means of "**%INCLUDE FP1MAINP**".

```
/*********************************************************************/
/* NAME             FP1MAINP                                         */
/* LANGUAGE         PL1                                              */
/* VERSION           811                                             */
/*                                                                   */
/*                 /-> FHS-CONTROL-INFO                              */
/*  FHS-MAIN-PAR -                                                   */
/*                 \-> FHS-MAP-PAR                                   */
/*                                                                   */
/*********************************************************************/

DCL 1  FHS_MAINP,
       2 FHS_MAIN_PAR                          CHAR(56),

       2 FHS_MAP_PAR,
         41 FHS_MAP_PAR_GENERAL                CHAR(160),

       2 FHS_MAP_PAR_OPTIONAL                  CHAR(60),

       2 FHS_EXIT_PAR,
         41 EXIT_IDENT_LEN                     BIN FIXED(31),
         41 EXIT_IDENT                         CHAR(8),
         41 EXIT_IN_OUT                        CHAR,
         41 EXIT_RET_INFO                      CHAR,
         41 FIL2                               CHAR(2),
         41 FIL4                               CHAR(4),
         41 EXIT_FLD_LEN                       BIN FIXED(31),
         41 EXIT_EFF_LEN                       BIN FIXED(31),
         41 EXIT_DATA                          CHAR(80);

DCL 1  FHS_CONTROL_INFO          DEFINED FHS_MAIN_PAR,
       4 FHS_MAIN_RC                           BIN FIXED(15),
       4 FIL6                                  CHAR(6),
       4 FHS_ERROR_INFO,
         41 ERROR_CATEGORY                     BIN FIXED(15),
         41 ERROR_REASON                       BIN FIXED(15),
       4 FIL7                                  CHAR(7),
       4 PRINTER_RETURNS_MSG,
         41 RETURN_MSG_TYPE                    CHAR,
         41 RETURN_BYTE1                       CHAR,
         41 RETURN_BYTE2                       CHAR,
         41 RETURN_STATUS_INFO                 CHAR(2),
```

```
        4 FHS_OUTPUT_INFO,
          41 FIL11                              CHAR(11),
          41 OUT_USER_AREA_TRUNCATION           CHAR,
          41 OUT_USER_AREA_LEN                  BIN FIXED(31),
        4 FHS_INPUT_INFO,
          41 FIL1                               CHAR,
          41 IN_PRINTER_RETURN_MSG              CHAR,
          41 IN_FIELD_DET                       CHAR,
          41 IN_MSG_NILS                        CHAR,
          41 IN_F_KEY                           BIN FIXED(15),
          41 IN_K_KEY                           BIN FIXED(15),
          41 IN_USER_AREA_LEN                   BIN FIXED(31),
          41 IN_MSG_LEN                         BIN FIXED(31);

DCL 1   FHS_MAP_GENERALS          DEFINED FHS_MAP_PAR_GENERAL,
        4 FHS_MAP_NAME                          CHAR(8),
        4 FHS_EXIT_MOD_NAME                     CHAR(8),
        4 FHS_MAPPING_METHOD                    CHAR(4),
        4 FHS_MODY_ATTRS                        CHAR,
        4 FHS_PARTIAL_MAP_OPT                   CHAR,
        4 FHS_MAP_PART                          CHAR,
        4 FHS_MAP_CURSOR_OPT                    CHAR,
        4 FIL4                                  CHAR(4),
        4 FIL6                                  CHAR(4),
        4 FHS_SERVICE_FUNCTION                  BIN FIXED(15),
        4 FHS_RESTART_OPT1                      CHAR,
        4 FHS_RESTART_OPT2                      CHAR,
        4 FHS_MAP_LIB_OPT                       CHAR,
        4 FHS_MAP_LIB_NAME                      CHAR(54),
        4 FHS_EXIT_LIB_OPT                      CHAR,
        4 FHS_EXIT_LIB_NAME                     CHAR(54),
        4 FHS_EXIT_FOR_OUTPUT                   CHAR,
        4 FHS_EXIT_FOR_INPUT                    CHAR,
        4 FHS_DESIRED_CCSNAME                   CHAR(8),
        4 FIL12                                 CHAR(4);

DCL 1   FHS_MAP_OPTIONS           DEFINED FHS_MAP_PAR_OPTIONAL,
        4 MAP_DEVICE_CLASS                      CHAR(4),
        4 MAP_PRINTER_CONTROL                   CHAR(4),
        4 FILLER                                CHAR(4),
        4 MAP_AUTO_TAB                          CHAR,
        4 MAP_EFF_LEN                           CHAR,
        4 MAP_POS_DET_CHAR                      CHAR,
        4 MAP_NEG_DET_CHAR                      CHAR,
        4 FIL8                                  CHAR(8),
        4 MAP_READ_METHOD                       CHAR(4),
        4 MAP_SCREEN_PRE_MOD                    CHAR,
        4 MAP_READ_NILS                         CHAR,
```

```
4 MAP_USE_ALL_ATTRS                    CHAR,
4 MAP_PRINTER_OPTION                   CHAR,
4 MAP_PRINTER_RETURN_BYTE1             CHAR,
4 MAP_PRINTER_RETURN_BYTE2             CHAR,
4 FIL1                                 CHAR,
4 MAP_HARDCOPY_OPTION                  CHAR,
4 FIL2                                 CHAR(2),
4 MAP_AUTO_HARDCOPY                    CHAR,
4 MAP_LOCK_KEYS                        CHAR,
4 MAP_CLEAR_OPTION                     CHAR,
4 MAP_BEL_OPTION                       CHAR,
4 MAP_PRINT_FORMAT_OPTION,
  41 MAP_PRINT_LINES                   CHAR,
  41 MAP_PRINT_COLUMNS                 CHAR,
  41 MAP_PRINT_PAPER                   CHAR,
  41 MAP_PRINT_FORM                    CHAR,
4 MAP_LIB_LOAD_OPTION,
  41 MAP_LIB_LOAD_MODE                 CHAR,
  41 MAP_LIB_LOAD_FILE                 CHAR,
4 MAP_HOLE_COLOR                       CHAR,
4 FIL11                                CHAR(11);
```

The individual data fields have the same meaning as in COBOL; see the section “The FHS-MAIN-PAR data structure” on page 303ff.

9.2.2 The FHS_INIT_PAR data structure

FHS_INIT_PAR is the name of the following data structure, which is copied into the program by means of **%INCLUDE FP1INITP**".

```
/*********************************************************************/
/* NAME             FP1INITP                                         */
/* LANGUAGE         PL1                                              */
/* VERSION          811                                              */
/*                                                                   */
/*                   FHSINITP FOR PLI1  V810                         */
/*                       DATA STRUCTURE FOR THE FHSINIT CALL         */
/*                                                                   */
/*********************************************************************/
DCL  1  FHS_INIT_PAR,
         2  FHS_INIT_PAR_GENERAL,
             41  FHS_I_O_AREA_LEN          BIN FIXED(31),
             41  FHS_RES_MAP_NO            BIN FIXED(15),
             41  FHS_MAP_NO                BIN FIXED(15),
             41  FIL7                      CHAR(7),
             41  FHS_ACCESS_METHOD         CHAR,

         2  FHS_MAPPING_DEFAULTS         CHAR(60),

         2  FHS_INIT_SYS_INFO            CHAR(80);

DCL  1  FHS_MAP_OPTIONS           DEFINED FHS_MAPPING_DEFAULTS,
             41  MAP_DEVICE_TYPE           CHAR(4),
             41  MAP_CONTROL_UNIT          CHAR(4),
             41  MAP_USER_AREA_LEN         BIN FIXED(31),
             41  MAP_AUTO_TAB              CHAR,
             41  MAP_EFF_LEN               CHAR,
             41  MAP_POS_DET_CHAR          CHAR,
             41  MAP_NEG_DET_CHAR          CHAR,
             41  FIL8                      CHAR(8),
             41  MAP_READ_METHOD           CHAR(4),
             41  MAP_SCREEN_PRE_MOD        CHAR,
             41  MAP_READ_NILS             CHAR,
             41  MAP_USE_ALL_ATTRS         CHAR,
             41  MAP_PRINTER_OPTION        CHAR,
             41  MAP_PRINTER_RETURN_BYTE1  CHAR,
             41  MAP_PRINTER_RETURN_BYTE2  CHAR,
             41  FIL1                      CHAR,
             41  MAP_HARDCOPY,
                 42 HARDCOPY_OPTION        CHAR,
                 42 CENTRAL_PRINT_ADDR     BIN FIXED(15),
```

```
            41  MAP_AUTO_HARDCOPY          CHAR,
            41  MAP_LOCK_KEYS              CHAR,
            41  MAP_CLEAR_OPTION           CHAR,
            41  MAP_BEL_OPTION             CHAR,
            41  MAP_PRINT_FORMAT_OPTION,
                42 MAP_PRINT_LINES         CHAR,
                42 MAP_PRINT_COLUMNS       CHAR,
                42 MAP_PRINT_PAPER         CHAR,
                42 MAP_PRINT_FORM          CHAR,
            41  MAP_LIB_LOAD_OPTION,
                42 MAP_LIB_LOAD_MODE       CHAR,
                42 MAP_LIB_LOAD_FILE       CHAR,
            41  MAP_HOLE_COLOR             CHAR,
            41  FIL11                      CHAR(11);

DCL  1  FHS_BS2000_INFO  DEFINED FHS_INIT_SYS_INFO,
            41  FIL25                 CHAR(25),
            41  FHS_MAP_LIB_OPT       CHAR,
            41  FHS_MAP_LIB_NAME      CHAR(54);
```

The individual data fields have the same meaning as in COBOL; see the section “The FHS-INIT-PAR data structure” on page 330ff.

9.2.3 The FHS_ATTR_PAR data structure

FHS_ATTR_PAR is the name of the following data structure, which is copied into the program by means of "**%INCLUDE FP1ATTRP**".

```
/*********************************************************************/
/*                                                                   */
/*       F H S A T T R P   V811                                      */
/*                                                                   */
/*            PL/I INCLUDE FOR FHSATTR CALL                          */
/*                                                                   */
/*********************************************************************/
DECLARE
01 FHS_ATTR_PAR,

    02  FHS_ATTR_PAR_BASIC,
        42 A_UPDATE_METHOD          CHAR(3) INIT('REP'),
        42 FILL01                   CHAR(5),
        42 A_PROT_LEVEL             CHAR(4),
        42 A_DISP_LEVEL             CHAR,
        42 FILL02                   CHAR(3),

    02  FHS_ATTR_PAR_OPTIONAL       CHAR(24);

DCL 01 FHS_ATTR_OPTIONS DEFINED FHS_ATTR_PAR_OPTIONAL,
        04 A_NO_HARDCOPY            CHAR,
        04 A_NUMERIC                CHAR,
        04 A_SIGNAL                 CHAR,
        04 A_ITALIC                 CHAR,
        04 FILL03                   CHAR(12),
        04 A_WIDE                   CHAR,
        04 A_TALL                   CHAR,
        04 FILL04                   CHAR(5),
        04 A_ASKIP                  CHAR;
```

The individual data fields have the same meaning as in COBOL; see the section "The FHS-ATTR-PAR data structure" on page 334ff.

9.2.4 The FHS_EXITMOD_PAR data structure

FHS_EXITMOD_PAR is the name of the following data structure, which is copied into the exit routine by means of “**INCLUDE FP1EXITP**”. FHSEXITP must be used as a PROC parameter.

```
/*********************************************************************
*  FHSEXITP   V811                                                   *
*                                                                    *
*                                                                    *
* DATA STRUCTURE FOR THE EXIT ROUTINE                                *
*                                                                    *
*********************************************************************/

DCL 01  FHS_EXITMOD_PAR,

    02  EXITMOD_PAR                  CHAR(108);

DCL 01 FHS_EXIT_PAR     DEFINED EXITMOD_PAR,
    02 EXIT_IDENT_LEN     BIN FIXED(31),
    02 EXIT_IDENT         CHAR(8),
    02 EXIT_IN_OUT        CHAR,
    02 EXIT_RET_INFO      CHAR,
    02 FIL1               CHAR,
    02 FIL2               CHAR(2),
    02 EXIT_FLD_LEN       BIN FIXED(31),
    02 EXIT_EFF_LEN       BIN FIXED(31),
    02 EXIT_DATA          CHAR(80);
```

The individual data fields have the same meaning as in COBOL; see the section “The FHS-EXITMOD-PAR data structure” on page 338ff.

9.2.5 The FHS_CCSN_PAR data structure

FHS_CCSN_PAR is the name of the following data structure, which is copied into the application program by means of **%INCLUDE FP1CCSNP;**. It is required for the FHSSERV call in order to determine the character set of a format.

```
/*********************************************************************/
/*                                                                   */
/*       F H S C C S N P   V811                                      */
/*                                                                   */
/*            PL/I INCLUDE FOR FHSSERV CALL                          */
/*                                                                   */
/*********************************************************************/
DECLARE
01 FHS_CCSN_PAR,

    02  FILL01                      CHAR(8),
    02  FHS_CCSN_INFO               CHAR(8),
    02  FILL02                      CHAR(16);
```

FHS enters the name of the character set in the field FHS_CCSN_INFO.

9.3 PL/I calls for TIAM

When using FHS, the TIAM calls WROUT and WRTRD can be issued in PL/I programs.

9.3.1 TIAM call WROUT

The TIAM call “CALL WROUT” serves to output formatted messages.

```
CALL WROUT(TIAM_CONTROL_INFO,transfer area,FHS_MAIN_PAR)
```

The parameters have the following meaning:

TIAM_CONTROL_INFO

PL/I data structure which controls the TIAM call. This data structure is shown in the “TIAM (TRANSDATA, BS2000)” User Guide.

transfer-area

Name of the data transfer area. The first field must be a variable which FHS supplies with the length of a message. This can be done as follows:

```
DECLARE 01 BINARY FIXED(31);
%INCLUDE FORMAT;
```

FHS_MAIN_PAR

controls formatting. Please refer to the section “TIAM calls” on page 343ff for a description of the entries you have to make in the individual fields and the return codes supplied by FHS.

9.3.2 TIAM call WRTRD

The TIAM call “CALL WRTRD” serves to input/output formatted messages.

```
CALL WRTRD(TIAM_CONTROL_INFO,transfer area1, transfer area 2, FHS_MAIN_PAR)
```

The parameters have the following meaning:

TIAM_CONTROL_INFO

PL/I data structure which controls the TIAM call. This Data structure is shown in the “TIAM (TRANSDATA, BS2000)” User Guide.

transfer area 1

Name of the output transfer area

transfer area 2

Name of the input transfer data. With #formats, the same data transfer area must be used for both input and output.

The first field of the two transfer areas must be a variable which FHS supplies with the length of the message. This can be done as follows:

```
DECLARE 01 BINARY FIXED(15);
%INCLUDE FORMAT;
```

FHS_MAIN_PAR

controls formatting. Refer to the section “TIAM calls” on page 343ff for a description of the entries you have to make in the individual fields and the return calls supplied by FHS.

9.4 FHS PL/I calls

The subprograms FHSCURS (Position cursor), FHSATTR (Modify +format attributes, FHSINIT (Initialize formatting) and FHSSERV (Initialize the data transfer area and determine name of character set) can be called in PL/I programs. o

9.4.1 FHSCURS

CALL FHSCURS serves to position the cursor in any unprotected or detectable field of a *format or +format. FHSCURS is called as follows:

```
CALL FHSCURS(FHS_MAIN_PAR,field-name);
```

“field-name” is the name of the field to which FHS is to position the cursor. FHS_MAIN_PAR contains the return codes of the FHSCURS call. For more details, see the section “CALL “FHSCURS”” on page 361.

9.4.2 FHSATTR

FHSATTR serves to modify the attribute fields of a field of an +format, thus modifying the field attributes. FHSATTR is called as follows:

```
CALL FHSATTR(FHS_CONTROL_INFO,FHS_ATTR_PAR,attribute-field);
```

“attribute-field” is the name of the attribute field that is to be modified. FHS_CONTROL_INFO is that part of the FHS_MAIN_PAR data structure containing the return codes. FHS_ATTR_PAR controls attribute modification. For more details see section “CALL “FHSATTR”” on page 363.

9.4.3 FHSINIT

The FHSINIT subprogram serves to initialize formatting and to specify the formats to be loaded upon opening formatting. The start parameters for #formats are defined with this call. CALL FHSINIT is necessary

– if you wish to work with formats that are to be loaded upon opening, or

– if you wish to use more than 100 different formats.

Furthermore, the FHSINIT call can be used to define your own formatting standard. This standard can be modified for every formatting operation.

If CALL FHSINIT is issued more than once, the data area FHS_INIT_PAR_GENERAL is not evaluated for any but the first call, since formatting is initiated by the first CALL “FHSINIT”. FHSINIT is called as follows:

```
CALL FHSINIT(FHS_CONTROL_INFO,FHS_INIT_PAR,area,[format-list]);
```

FHS_CONTROL_INFO is part of the FHS_MAIN_PAR data structure (see page 459). FHS_INIT_PAR is shown on page 462. “area” may be any area, it must be specified for compatibility reasons. “format-list” is the name of a list containing the names of the formats that are to be loaded upon opening formatting.

Please refer to the section “CALL “FHSINIT”” on page 366 for the entries to be made in the individual fields. Except for the underscores, the field names are identical with the COBOL field names.

9.4.4 FHSSERV

The FHSSERV subprogram enables you to use two FHS service functions:

– ’Initialization of the Data Transfer Area for #formats
– ’Determine name of character set’

9.4.4.1 Initialization of the data transfer area

All field attributes are set according to their default values in the format. Neither the global attributes (except for ’Formatting acknowledgment) nor the field contents are updated. It is thus possible to reset to the initial status data transfer areas that have already been supplied with data. FHSSERV is called as follows:

```
CALL FHSSERV(FHS_MAIN_PAR,transfer-area);
```

The FHS_MAIN_PAR data structure is shown on page 459. “transfer-area” is the name of the transfer-area without the preceding length field.

Please refer to the section “CALL “FHSSERV”” on page 370 for the entries to be made in the individual fields. Except for the underscores, the field names are identical with the COBOL field names.

9.4.4.2 Determine name of character set

In this case, the format specified in FHS_MAIN_PAR is loaded and the name of the relevant character set is entered into the FHS_CCSN_INFO field of the FHS_CCSN_PAR data structure. For this function FHS_CCSN_PAR is called as follows:

```
CALL FHSSERV(FHS_MAIN_PAR,FHS_CCSN_PAR);
```

The FHS_MAIN_PAR data structure is described on page 459; FHS_CCSN_PAR can be found on page 466.

9.4.4.3 Unload format

The format specified in FHS_MAIN_PAR is unloaded and can be replaced by a modified format.

FHSSERV is called as follows for this function:

```
CALL FHSSERV(FHSMAINPAR,transfer-area)
```

The FHS_MAIN_PAR data structure is described on page 459. “transfer-area” is the name of the transfer-area without the preceding length field.

Please refer to the section “Unload format” on page 374 for the entries to be made in the individual fields. Except for the underscores, the field names are identical with the COBOL field names.

9.4.4.4 Dynamic retrieval of information on the structure of the addressing aid for #formats

The structure of the addressing aid for the format specified in FHS_MAIN_PAR is returned in the transfer area

FHSSERV is called as follows for this function:

```
CALL FHSSERV(FHSMAINPAR,transfer-area)
```

The FHS_MAIN_PAR data structure is described on page 459. “transfer-area” is the name of the transfer-area without the preceding length field.

Please refer to the section “Unload format” on page 374 for the entries to be made in the individual fields. Except for the underscores, the field names are identical with the COBOL field names.

9.5 Attribute modification

9.5.1 List of attribute values FHS_ATTRIBUTE_VALUES

FHS_ATTRIBUTE_VALUES is the following list of attribute values which is copied into the application program by means of the PL/I statement "**%INCLUDE FP1AVAL**". It generates the symbolically addressable attribute values for the global attributes and field attributes of the data transfer area with separate attribute blocks and field contents.

```
/*********************************************************************/
/* NAME             FP1AVAL                                          */
/* LANGUAGE         PL1                                              */
/* VERSION          811                                              */
/*                                                                   */
/*           THIS INCLUDE GENERATES ATTRIBUTE VALUES                 */
/*           FOR MAPS WITH EXTENDED USER AREA (EUA)                  */
/*                                                                   */
/*********************************************************************/

DECLARE (

/*      GLOBAL ATTRIBUTE VALUES (CHARACTERS)   */
/*      -------------------------------------  */

/* DEFAULT VALUES */
         GA_DEFAULT           INIT(, ,),

/* FORMATTING INDICATORS */
   /* FIELDS MODIFICATION */
         GA_MODIFIED          INIT(,Y‘),
         GA_NOT_MODIFIED      INIT(, ,),
   /* FIELDS DETECTION */
         GA_DETECTED          INIT(,Y‘),
         GA_NOT_DETECTED      INIT(, ,),
   /* FIELDS VALIDATION */
         GA_VALID             INIT(,V‘),
         GA_NOT_VALID         INIT(, ,),
   /* FIELDS UNDEFINED  */
         GA_UNDEFINED         INIT(,Y‘),
         GA_NOT_UNDEFINED     INIT(, ,),
```

```
/* INPUT IDENTIFICATION */
   /* INPUT KEY CLASS */
         GA_INPUT_KEY          INIT(,I‘),
         GA_F_KEY              INIT(,F‘),
         GA_K_KEY              INIT(,K‘),
         GA_POS_RM             INIT(,P‘),
         GA_NEG_RM             INIT(,N‘),
         GA_INPUT_NONE         INIT(, ,),

/* DEVICE CONTROLS */
   /* INIT CONTROL */
         GA_NO_INIT            INIT(,N‘),
         GA_FIRST_INIT         INIT(,F‘),
         GA_LAST_INIT          INIT(,L‘),
         GA_BOTH_INIT          INIT(,B‘),
   /* TABULATOR CONTROL */
         GA_AUTO_TAB           INIT(,A‘),
         GA_NO_AUTO_TAB        INIT(,N‘),
   /* FUNCTION LOCK */
         GA_KEYLOCK            INIT(,K‘),
   /* VMI CONTROL */
         GA_VMI_1              INIT(,1‘),
         GA_VMI_2              INIT(,2‘),
         GA_VMI_3              INIT(,3‘),
   /* HMI CONTROL */
         GA_HMI_1              INIT(,1‘),
         GA_HMI_2              INIT(,2‘),
         GA_HMI_3              INIT(,3‘),

/* OUTPUT CONTROLS */
   /* CYCLE CONTROL */
         GA_CLOSE              INIT(,C‘),
   /* COPY CONTROL */
         GA_HARDCOPY_GEN       INIT(,H‘),
         GA_HARDCOPY_LOC       INIT(,L‘),
   /* ALARM CONTROL */
         GA_ALARM              INIT(,A‘),
   /* HOLE COLOR  */
         GA_NO_COLOR           INIT(,U‘),
         GA_WHITE_HOLE         INIT(,W‘),
         GA_GREY_HOLE          INIT(,G‘),
```

```
/* FORMATTING CONTROLS */
    /* DISPLAY SELECTION */
         GA_BOXB              INIT(,B‘),
         GA_BOXL              INIT(,C‘),
         GA_KEB               INIT(,K‘),
         GA_KEL               INIT(,L‘),
    /* LEVEL SELECTION */
         GA_LEVEL_1           INIT(,1‘),
         GA_LEVEL_2           INIT(,2‘),
         GA_LEVEL_3           INIT(,3‘),
         GA_LEVEL_P           INIT(,P‘),
    /* OUTPUT MODE */
         GA_RDIF              INIT(,R‘),
    /* CURSOR CONTROL */
         GA_FIELD_CURSOR      INIT(,F‘),
         GA_EDIT_CURSOR       INIT(,E‘),
         GA_REL_CURSOR        INIT(,R‘),
    /* USER EXIT CONTROL   */
         GA_NO_UEXIT          INIT(,N‘),
         GA_OUT_UEXIT         INIT(,O‘),
         GA_IN_UEXIT          INIT(,I‘),
         GA_BOTH_UEXIT        INIT(,B‘),

 /*       FIELD ATTRIBUTE VALUES (CHARACTERS)   */
 /*       ------------------------------------   */

 /* DEFAULT VALUES */
         FA_DEFAULT           INIT(, ,),

 /* BASIC ATTRIBUTES */
    /* INPUT STATE / INPUT STATE ACT */
         FA_MODIFIED          INIT(,M‘),
         FA_CLEARED           INIT(,C‘),
         FA_DETECTED          INIT(,D‘),
         FA_UNDEFINED         INIT(,U‘),
         FA_NOT_TOUCHED       INIT(, ,),
    /* EDIT STATE */
         FA_VALID             INIT(,V‘),
         FA_INVALID           INIT(,I‘),
         FA_MUST_ERROR        INIT(,M‘),
         FA_NOT_CHECKED       INIT(, ,),
    /* OUTPUT CONTROL      */
         FA_OUTPUT_INIT       INIT(,I‘),
         FA_OUTPUT_DATA       INIT(,D‘),
         FA_OUTPUT_UNDEFINED  INIT(,U‘),
```

```
/* FIELD INPUT */
   /* INPUT CONTROL */
         FA_NORMAL_IN          INIT(,N'),
         FA_MUST_IN            INIT(,M'),
         FA_POTMUST_IN         INIT(,P'),
         FA_AUTORET_IN         INIT(,A'),
   /* PROTECTION */
         FA_UNPROTECTED        INIT(,U'),
         FA_PROTECTED          INIT(,P'),
         FA_ASKIP              INIT(,A'),
         FA_DETECTABLE         INIT(,D'),

/* DISPLAY CONTROL */
   /* INTENSITY */
         FA_HIGH_INTENSITY     INIT(,H'),
         FA_NORMAL_INTENSITY   INIT(,N'),
   /* VISIBILITY */
         FA_VISIBLE            INIT(,V'),
         FA_SIGNALING          INIT(,S'),
         FA_INVISIBLE          INIT(,I'),
   /* UNDERLINE */
         FA_UNDERLINED         INIT(,Y'),
         FA_NOT_UNDERLINED     INIT(,N'),
   /* INVERSE */
         FA_INVERSE            INIT(,Y'),
         FA_NOT_INVERSE        INIT(,N'),

/* COLOUR */
         FA_RED                INIT(,1'),
         FA_GREEN              INIT(,2'),
         FA_YELLOW             INIT(,3'),
         FA_BLUE               INIT(,4'),
         FA_MAGENTA            INIT(,5'),
         FA_CYAN               INIT(,6'),
         FA_WHITE              INIT(,7'),
         FA_NO_COLOR           INIT(,N'),

/* CURSOR */
         FA_CURSOR             INIT(,Y'),
         FA_HOLD_CURSOR        INIT(,H'),
         FA_NO_CURSOR          INIT(,N')

                              ) CHARACTER(1) STATIC(CONSTANT);
```

9.5.2 The FHS_ATTRIBUTE_MOVE data structure

This section describes the attribute modifications for +formats. It does not apply to #formats unless the field attribute group 'attribute combination' is used (see page 62ff).

FHS offers the PL/I programmer two possibilities for modifying the field attributes of the format fields:

– by means of "FHSATTR" (see page 469ff) or

– by means of the COPY element **FHS_ATTRIBUTE_MOVE**. If a suitable attribute combination exists in the data structure, you can enter it in the desired attribute field by means of a simple statement.

FHS_ATTRIBUTE_MOVE is the name of the following data structure, which is copied into the application program by means of "**%INCLUDE FP1ATTRM**".

```
/*********************************************************************/
/* NAME             FP1ATTRM                                         */
/* LANGUAGE         PL1                                              */
/* VERSION          811                                              */
/*                                                                   */
/* DATA STRUCTURE FOR THE ATTRIBUTE MOVE                             */
/*                                                                   */
/*********************************************************************/
DECLARE
01 FHS_ATTRIBUTE_MOVE,
/*********************************************************************/

/* KCALPH   ** UNPROT,BRT,PRINT                                      */
        02 KYALPH                    BIN FIXED(31) INIT(20512),
        02 KXALPH,
              03 KCALPH              BIN FIXED INIT(20512),
              03 FILLER              PIC ,99‘,
/* KCNUME   ** UNPROT,BRT,NUM                                        */
        02 KYCNUME                   BIN FIXED(31) INIT(21024),
        02 KXCNUME,
              03 FILLER              PIC ,99‘,
              03 KCNUME              BIN FIXED(15) INIT(21024),
/* KCPROT   ** PROT,NORM                                             */
        02 KCPROT                    BIN FIXED(15) INIT(4360),
/* KCUNPR   ** UNPROT,BRT                                            */
        02 KYCUNPR                   BIN FIXED(31) INIT(20512),
        02 KXCUNPR,
              03 FILLER              PIC ,99‘,
              03 KCUNPR              BIN FIXED(15) INIT(20512),
```

```
/* KCNINT   ** UNPROT,NORM                                          */
        02 KYCNINT                    BIN FIXED(15) INIT(20488),
        02 KXCNINT,
              03 FILLER               PIC ,99',
              03 KCNINT               BIN FIXED(15) INIT(20488),
/* KCDINT   ** UNPROT,DRK                                           */
        02 KYCDINT                    BIN FIXED(15) INIT(20488),
        02 KXCDINT,
              03 FILLER               PIC ,99',
              03 KCDINT               BIN FIXED(15) INIT(20488),
/* KCHINT   ** UNPROT,DRK                                           */
        02 KYCHINT                    BIN FIXED(15) INIT(20512),
        02 KXCHINT,
              03 FILLER               PIC ,99',
              03 KCHINT               BIN FIXED(15) INIT(20512),
/* KCITAL   ** UNPROT,BRT,ITAL                                      */
        02 KYCITAL                    BIN FIXED(15) INIT(20514),
        02 KXCITAL,
              03 FILLER               PIC ,99',
              03 KCITAL               BIN FIXED(15) INIT(20514),
/* KCSIGN   ** UNPROT,BRT,SIGN                                      */
        02 KYCSIGN                    BIN FIXED(15) INIT(20513),
        02 KXCSIGN,
              03 FILLER               PIC ,99',
              03 KCSIGN               BIN FIXED(15) INIT(20513),
/* KCDETE   ** PROT,BRT,DET                                         */
        02 KCDETE                     BIN FIXED(31) INIT(6432),
/* KCPREM   ** FSET,BRT                                             */
        02 KCPREM                     BIN FIXED(31) INIT(5152),
/* KCAUN    ** UNPROT,NORM                                          */
        02 KYCAUN                     BIN FIXED(31) INIT(20488),
        02 KXCAUN,
              03 FILLER               PIC ,99',
              03 KCAUN                BIN FIXED(15) INIT(20488),
/* KCNUN    ** UNPROT,NORM,NUM                                      */
        02 KYCNUN                     BIN FIXED(31) INIT(21000),
        02 KXCNUN,
              03 FILLER               PIC ,99',
              03 KCNUN                BIN FIXED(15) INIT(21000),
/* KCAPN    ** PROT,NORM                                            */
        02 KCAPN                      BIN FIXED(31) INIT(4360),
/* KCNPN    ** PROT,NORM,NUM                                        */
        02 KCNPN                      BIN FIXED(31) INIT(4872),
/* KCAUD    ** UNPROT,DRK                                           */
        02 KYCAUD                     BIN FIXED(31) INIT(20484),
        02 KXCAUD,
              03 FILLER               PIC ,99',
              03 KCAUD                BIN FIXED(15) INIT(20484),
```

```
/* KCNUD    ** UNPROT,DRK,NUM                                          */
        02 KYCNUD                    BIN FIXED(31) INIT(20996),
        02 KXCNUD,
             03 FILLER               PIC ,99‘,
             03 KCNUN                BIN FIXED(15) INIT(20996),
/* KCAPD    ** PROT,DRK                                               */
        02 KCAPD                     BIN FIXED(31) INIT(4356),
/* KCNPD    ** PROT,DRK,NUM                                           */
        02 KCNPD                     BIN FIXED(31) INIT(4868),
/* KCAUH    ** UNPROT,BRT                                             */
        02 KYCAUH                    BIN FIXED(31) INIT(20512),
        02 KXCAUH,
             03 FILLER               PIC ,99‘,
             03 KCAUH                BIN FIXED(15) INIT(20512),
/* KCNUH    ** UNPROT,BRT,NUM                                         */
        02 KYCNUH                    BIN FIXED(31) INIT(21024),
        02 KXCNUH,
             03 FILLER               PIC ,99‘,
             03 KCNUH                BIN FIXED(15) INIT(21024),
/* KCAPH    ** PROT,BRT                                               */
        02 KCAPH                     BIN FIXED(31) INIT(4384),
/* KCNPH    ** PROT,BRT,NUM                                           */
        02 KCNPH                     BIN FIXED(31) INIT(4896),
/* KCAUI    ** UNPROT,BRT,ITAL                                        */
        02 KYCAUI                    BIN FIXED(31) INIT(20514),
        02 KXCAUI,
             03 FILLER               PIC ,99‘,
             03 KCAUI                BIN FIXED(15) INIT(20514),
/* KCNUI    ** UNPROT,BRT,ITAL,NUM                                    */
        02 KYCNUI                    BIN FIXED(31) INIT(21026),
        02 KXCNUI,
             03 FILLER               PIC ,99‘,
             03 KCNUI                BIN FIXED(15) INIT(21026),
/* KCAPI    ** PROT,NORM,ITAL                                         */
        02 KCAPI                     BIN FIXED(31) INIT(4362),
/* KCNPI    ** PROT,NORM,ITAL,NUM                                     */
        02 KCNPI                     BIN FIXED(31) INIT(4874),
/* KCAUS    ** UNPROT,BRT,SIGN                                        */
        02 KYCAUS                    BIN FIXED(31) INIT(20513),
        02 KXCAUS,
             03 FILLER               PIC ,99‘,
             03 KCAUS                BIN FIXED(15) INIT(20513),
/* KCNUS    ** UNPROT,BRT,SIGN,NUM                                    */
        02 KYCNUS                    BIN FIXED(31) INIT(21025),
        02 KXCNUS,
             03 FILLER               PIC ,99‘,
             03 KCNUS                BIN FIXED(15) INIT(21025),
```

```
/* KCAPS    ** PROT,NORM,SIGN                                          */
       02 KCAPS                      BIN FIXED(31) INIT(4361),
/* KCNPS    ** PROT,NORM,SIGN,NUM                                      */
       02 KCNPS                      BIN FIXED(31) INIT(4873),
/* KCAUND   ** UNPROT,NORM,DET                                         */
       02 KYCAUND                    BIN FIXED(31) INIT(22536),
       02 KXCAUND,
            03 FILLER                PIC ,99‘,
            03 KCAUND                BIN FIXED(15) INIT(22536),
/* KCNUND   ** UNPROT,NORM,DET                                         */
       02 KYCNUND                    BIN FIXED(31) INIT(22536),
       02 KXCNUND,
            03 FILLER                PIC ,99‘,
            03 KCNUND                BIN FIXED(15) INIT(22536),
/* KCAPND   ** PROT,NORM,DET                                           */
       02 KCAPND                     BIN FIXED(31) INIT(6408),
/* KCNPND   ** PROT,NORM,DET,NUM                                       */
       02 KCNPND                     BIN FIXED(31) INIT(6920),
/* KCAUHD   ** UNPROT,BRT,DET                                          */
       02 KYCAUHD                    BIN FIXED(31) INIT(22560),
       02 KXCAUHD,
            03 FILLER                PIC ,99‘,
            03 KCAUHD                BIN FIXED(15) INIT(22560),
/* KCNUHD   ** UNPROT,BRT,DET                                          */
       02 KYCNUHD                    BIN FIXED(31) INIT(22560),
       02 KXCNUHD,
            03 FILLER                PIC ,99‘,
            03 KCNUND                BIN FIXED(15) INIT(22560),
/* KCAPHD   ** PROT,BRT,DET                                            */
       02 KCAPHD                     BIN FIXED(31) INIT(6432),
/* KCNPHD   ** PROT,BRT,DET,NUM                                        */
       02 KCNPHD                     BIN FIXED(31) INIT(6944),
/* KCAUID   ** UNPROT,BRT,DET,ITAL                                     */
       02 KYCAUID                    BIN FIXED(31) INIT(22562),
       02 KXCAUID,
            03 FILLER                PIC ,99‘,
            03 KCAUID                BIN FIXED(15) INIT(22562),
/* KCNUID   ** UNPROT,BRT,DET,ITAL                                     */
       02 KYCNUID                    BIN FIXED(31) INIT(22562),
       02 KXCNUID,
            03 FILLER                PIC ,99‘,
            03 KCNUID                BIN FIXED(15) INIT(22562),
/* KCAPID   ** PROT,NORM,DET,ITAL                                      */
       02 KCAPID                     BIN FIXED(31) INIT(6410),
/* KCNPID   ** PROT,NORM,DET,ITAL,NUM                                  */
       02 KCNPID                     BIN FIXED(31) INIT(6922),
```

```
/* KCAUSD   ** UNPROT,BRT,DET                                          */
        02 KYCAUSD                    BIN FIXED(31) INIT(22560),
        02 KXCAUSD,
              03 FILLER               PIC ,99‘,
              03 KCAUSD               BIN FIXED(15) INIT(22560),
/* KCNUSD   ** UNPROT,BRT,DET                                          */
        02 KYCNUSD                    BIN FIXED(31) INIT(22560),
        02 KXCNUSD,
              03 FILLER               PIC ,99‘,
              03 KCNUSD               BIN FIXED(15) INIT(22560),
/* KCAPSD   ** PROT,NORM,DET                                           */
        02 KCAPSD                     BIN FIXED(31) INIT(6408),
/* KCNPSD   ** PROT,NORM,DET,NUM                                       */
        02 KCNPSD                     BIN FIXED(31) INIT(6920),
/* KCAUNP   ** FSET,NORM                                               */
        02 KCAUNP                     BIN FIXED(31) INIT(5128),
/* KCNUNP   ** FSET,NORM,NUM                                           */
        02 KCNUNP                     BIN FIXED(31) INIT(5640),
/* KCAPNP   ** PROTRET,NORM                                            */
        02 KYCAPNP                    BIN FIXED(31) INIT(12296),
        02 KXCAPNP,
              03 FILLER               PIC ,99‘,
              03 KCAPNP               BIN FIXED(15) INIT(12296),
/* KCNPNP   ** PROTRET,NORM,NUM                                        */
        02 KYCNPNP                    BIN FIXED(31) INIT(12808),
        02 KXCNPNP,
              03 FILLER               PIC ,99‘,
              03 KCNPNP               BIN FIXED(15) INIT(12808),
/* KCAUHP   ** FSET,BRT                                                */
        02 KCAUHP                     BIN FIXED(31) INIT(5152),
/* KCNUHP   ** FSET,BRT,NUM                                            */
        02 KCNUHP                     BIN FIXED(31) INIT(5664),
/* KCAPHP   ** PROTRET,BRT                                             */
        02 KYCAPHP                    BIN FIXED(31) INIT(12320),
        02 KXCAPHP,
              03 FILLER               PIC ,99‘,
              03 KCAPHP               BIN FIXED(15) INIT(12320),
/* KCNPHP   ** PROTRET,BRT,NUM                                         */
        02 KYCNPHP                    BIN FIXED(31) INIT(12832),
        02 KXCNPHP,
              03 FILLER               PIC ,99‘,
              03 KCNPHP               BIN FIXED(15) INIT(12832);
/*********************************************************************
```

9.6 Compiler-dependent constraints

FHS PL/I programs in XS format can only be generated as of TIAM version V11.0. The following compiler-dependent constraints therefore apply:

– When using an earlier TIAM version than 11.0, the compiler option OPTIONS=NOXS (=default option) must be specified for the compilation of the PL/I program.

– When using TIAM version V11.0, it is also possible to specify the compiler option OPTIONS=XS for the compilation of the PL/I program.

9.7 PL/I example

The sample program displays the format shown below on the screen and reads the subsequently made entries. The format has two input fields and one output field.

```
SCREEN FOR TEST PL/I INCLUDE MEMBERS FHS V80

INPUT FIELD 1 : @@@@@@@@@@
INPUT FIELD 2 : @@@@@@@@@@
OUTPUT FIELD  : @@@@@@@@@@
```

The example is implemented by the following PL/I program:

```
PGMPL1: PROC OPTIONS(MAIN);
       DCL WRTRD ENTRY EXTERNAL OPTIONS (ASSEMBLER);
  DCL    1 IOAREA,
         19 DTLNGT   BIN FIXED(15),
         %INCLUDE PGM.LIB(PL1);
  DCL   IOPR BIT(528) DEFINED IOAREA;
         %INCLUDE PLI1.071(FP1MAINP);
         %INCLUDE PLI1.071(TIP1INFO);
      IOPR = (528)'0'B;
      FIELDOUT='MESSFRPGM ';
      FHS_MAP_NAME = 'PL1';
      FHS_RESTART_OPT1 ='Y';
      FHS_MAP_LIB_OPT='Y';
      FHS_MAP_LIB_NAME='PGM.LIB';
      EDIT_OUT.EDIT_MODE='F';
      DISPLAY ('START OF PROGRAM');
      CALL WRTRD(TIAM_CONTROL_INFO,DTLNGT,DTLNGT,FHS_MAIN_PAR);
      DISPLAY ('TIAMRC    :'||CHAR(TIAM_RC));
      DISPLAY ('FHSMAINRC :'||CHAR(FHS_MAIN_RC));
      DISPLAY ('ERCATEGO  :'||CHAR(ERROR_CATEGORY));
      DISPLAY ('ERREASON  :'||CHAR(ERROR_REASON));
      DISPLAY ('CONTENT OF THE FIELDS :');
      DISPLAY ('FIELD1    :'||FIELD1);
      DISPLAY ('FIELD2    :'||FIELD2);
      DISPLAY ('FIELDOUT  :'||FIELDOUT);
      END;
```

10 Utility routines and print routines

This section describes the utility routine MAPPRINT, the print routine for formats, and creation of a user-own code table module.

10.1 Printing out formats, MAPPRINT

MAPPRINT retrieves a format definition from the format application file and outputs the format to SYSLST. In addition, the format is displayed on the screen for several seconds. The following figure shows how format ADDRES2, for example, is printed out.

```
MFHSFORM      MAPNAME:  ADRESS2       TERMINAL TYPE:  8161          DATE: 11/25/86
    1   5   10   15  (1)  25   30   35   40 (2) 45   50   55   60   65   70   75   80
   -+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+-
 1 | PLEASE ENTER YOUR ADRESS                                                       |
 2 |                                                                                |
 3 |                                                                                |
 4 |NAME: @@@@@@@@@@@@@@@@ FIRST NAME: @@@@@@@@@@@@@@@@@@@@@@                      |
 5 |      (3)                                                                       |
 6 |STREET: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
 7 |ZIP @@@@@ CITY @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
 8 |TELEPHONE:  @@@@@/@@@@@@@@                                                      |
 9 |                                                                                |
10 | ADRESS 2:                                                                      |
11 |                                                                                |
12 |STREET:@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
13 |ZIP @@@@@ CITY @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
14 |TELEPHONE  @@@@@/@@@@@@@@                                                       |
15 |                                                                                |
16 |                                                                                |
17 |                                                                                |
MAP/S USED:  ADRESS2 (4)    MAPLIB:  BIBL.GGS  (5)

SUB-FORMAT LEVEL1     SUBAD (6)
```

Printout generated by MAPPRINT

Key

1 Name of the format printed out

2 Terminal for which the format was generated

3 Display of the formats on the screen by MAPPRINT.

Variable fields are also displayed on the screen by the character @. With formats generated for the 9763 Data Display Terminal, data fields can be displayed with characters from user-generated character sets. The variable fields are then filled with the character defined for code position X'7C'.

4 Format definition used

5 Format application file used

6 Format definition used, subformat (1st level)

Control statements for MAPPRINT

Operation	Operands	Meaning
PRINT	formatname	Print out a format
MAPLIB=	{ F.MAPLIB / libname }	Specify the format application file
END		End statement

Meaning of the operands

formatname Name of the format to be printed out

libname Name of the format application file (up to 54 characters) in accordance with BS2000 conventions

Note

If there are formats with the same format name in different format application files, MAPPRINT always prints the format from the first format application file specified with MAPLIB=. Each MAPLIB statement overwrites any preceding MAPLIB statement.

If no MAPLIB statement has been entered, MAPPRINT will search in the file F.MAPLIB.

MAPPRINT reads the control statements from SYSDTA and can be called in interactive or batch mode.

In interactive mode MAPPRINT responds as follows:

```
MAPPRINT PROGRAM VERSION XXX
ENTER CONTROL COMMAND
*
```

```
XXX is the version number of MAPPRINT
```

Note

In the case of output messages generated for a 9763 Data Display Terminal, MAPPRINT ignores all statements relating to character sets and the field attributes concerning character sets and colors for output on SYSLST. The screen dimension specifications are evaluated.

Error messages:

The following error message is output on SYSOUT in the event of an incorrect control statement:

```
3410 61 characters of the faulty control statement
```

The following error message is output on SYSOUT or, in batch mode, SYSLST:

```
ERROR DURING FHS OPEN; RETURNCODE:04/140X
```

The program is then terminated. For the meaning of the return code see MOMAP return codes.

The following error message is output on SYSOUT and SYSLST in the event of a serious format error:

```
MAP formatname:SEVERE ERROR S.R.C.XXXX – NO PRINT
```

The format is not printed out.

formatname Name of the format

XXXX You will find the S.R.C. return code in the table “Secondary return codes after execution of the MCMAP macro”, see page 513ff.

When formatting formats for a 9763 Data Display Terminal with character sets the messages may be very large. If a message is greater than 8 Kbytes, the format is not displayed on the data display terminal but is instead output on SYSLST.

The error message then has the following format:

```
MAP formatname:SEVERE ERROR S.R.C.200C
OUTPUT MESSAGE >  8 K; MAP DISPLAY: PRINTER
```

If a format cannot be displayed on the data display terminal because:

– the terminal type in the format is incompatible with the type of terminal on which the format is supposed to be displayed, or

– the format was generated for a printer,

MAPPRINT issues the following message on the data display terminal:

```
MAP formatname: NO DISPLAY ON TERMINAL/LOCAL PRINTER
```

However, the format is output on SYSLST.

Note

Formats are generally output in their full width. In the case of formats generated for the 9763 Data Display Terminal with the screen format 27 x 132 and formats for printers having more than 80 print positions per line this can result in a corrupted printout (line feed forced by the system printer). It is therefore advisable to assign a file on SYSLST and to use the PRINT command with an appropriate character set to output this file on a printer.

If the MAPPRINT work area is not sufficiently large for editing printer formats the following message:

```
NOTE:SCREEN DISPLAY TRUNCATED
```

is output to SYSLST along with the format.

10.2 Print routines for formats

Using the MFHSFORM and MFHSFORR modules you can edit formats for printing and output them on a (high-speed) printer in parallel with the dialog taking place on a data display terminal.

The two modules differ only in terms of execute authorization. MFHSFORM is coded as non-shareable; MFHSFORR is shareable. In other words, applications written in “share mode” must use the MFHSFORR module.

The input consists of a message as supplied in the IOAREA by FHS during formatting, or as stored in the IOAREA after coming from the data display terminal. When a message is output, it is written to SYSLST with WRLST.

The application is responsible for the uniqueness of the coded character set of the resulting (system) file. FHS will output a return code if the current coded character set of the format (7-/8 bit or UTF-16) does not match the coded character set of the file currently assigned to SYSLST.

In output messages generated for a 9763 Data Display Terminal and logged on SYSLST, MAPPRINT ignores all specifications relating to character sets and the field attributes concerning character sets and colors. The screen dimension specifications are evaluated.

Note

Formats are generally output in their full width. In the case of formats generated for the 9763 Data Display Terminal with the screen format 27 x 132 and formats for printers having more than 80 print positions per line this can result in a corrupted printout (line feed forced by the system printer). It is therefore advisable to assign a file on SYSLST and to use the PRINT command with an appropriate character set to output this file on a printer. Formats having 132 columns can only be printed out correctly if MFHSFORM is activated immediately upon commencement of the dialog.

The MFHSFORM module is loaded with the MLINK macro (MFHSFORR with the MLINR macro) from the file assigned in the command

```
/SET-FILE-LINK LINK=MROUTLIB,FILE-NAME=libraryname
```

If no such FILE assignment has been made, loading takes place from the system file TASKLIB. This can be assigned by means of the command

```
/SET-TASKLIB LIBRARY=libraryname
```

Without any assignment in force, loading is attempted from the user file TASKLIB and finally from the file $.TASKLIB.

10.2.1 Load MFHSFORM, MLINK macro

Operation	Operands
MLINK	{name / (r)} [,prefix] [,mode]

Meaning of the operands:

name Name of a control block, up to 8 characters long, that was generated with the MDCBL macro.

(r) Decimal number or name, up to 8 characters long, of a register that must be loaded with the start address of a control block generated with the MDCBL macro

prefix Prefix up to 3 characters long.
If you do not specify a prefix, the default value MLI is assumed. MLINK generates the symbolic names prefix.00001, prefix.00002, prefix.00003,.... and RDT00001.

mode Specifies how the module MFHSFORM is to be loaded.

V If mode “V” is specified, MFHSFORM must be linked to the application program which then sets up the connection to MFHSFORM.
If no mode is entered, the module MFHSFORM is loaded by means of the MLINK macro. If the MFHSFORM module cannot be loaded, ...MRCF=X’0004’ and ....MSRC=X’1404’ are returned.
In either case the entry point is supplied in register 1; otherwise, register 1 cannot be used as the entry point.

The MLINK macro can only be used once in the program; any further MLINK calls are rejected. MLINK uses registers 1, 14 and 15. If necessary the user should save them before the macro call.

MLINK has XS capability.

10.2.2 Parameters and register entries

You can enter values in the registers either directly or via a parameter area.

Register entries, direct

Register	Contents
R13	Address of a save area, length of area: 18 full words
R14	Return address
R15	Entry point in the module MFHSFORM as supplied by the MLINK macro in register 1.
R1	Address of the FHSFORM parameter area
R0	Control information, length 4 bytes Byte 1-2 Pagination X‘0000‘ Internal pagination (1,2,3...) X‘nnnC‘ External pagination; the packed decimal number contained in these two bytes is used as the page number, e.g. X' 123C' for page 123. Byte 3: Specification of the access method X‘04‘ The message to be edited was generated for TIAM. X‘06‘ The message to be edited was generated for DCAM. Byte 4: Bit 2^0: Logging control =0 Header line is logged with internal and external pagination on every page. =1 No header line is logged. Bits 2^1 and 2^2 are reserved. Bit 2^3: Message type =0 Output message for data display terminal or printer =1 Input message from data display terminal Bit 2^4: Print control (for input messages only) =0 Work area is cleared prior to processing of the input message. =1 Work area is not cleared prior to processing of the input message. Bit 2^5: Output on a printer =0 Edited message is output on a printer. =1 Edited message is not output on a printer (for input and output messages). Bit 2^6: Switch for lowercase/uppercase letters =0 Lowercase letters are not converted =1 Output in uppercase letters Bit 2^7: Reserved

Note

The module MFHSFORM requires the work area in order to store the print-edited message. You should therefore ensure that this area is sufficiently large, i.e. at least 2 Kbytes or 4 Kbytes for Unicode formats respectively.
No printer control characters are stored in the work area.
Data fields with the FHS attribute NOPRINT are not output on the printer. The print control specification “Do Not Clear Work Area” is only evaluated if positioning sequences are found in the input message, e.g. read mode = read modified. If any are found, the printedited format in the work area will not be deleted if bit 2^4 is set.
The data supplied by the data display terminal is incorporated into the format stored in the work area in accordance with the positioning sequences (field addresses). This enables a dialog step (input, output) on the terminal to be logged on the printer. Before the start of the next dialog step, i.e. prior to processing of the next message, the work area is always cleared.

Register entries, via a parameter area

```
Parameter area:
DC  A(save area)                        for R13
DC  A(return address)                   for R14
DC  A(entry point in MFHSFORM)          for R15
DC  A(0)  CONTROL INFORMATION           for R0
DC  A(FHSFORM parameter area)           for R1

FHSFORM parameter area:
DC  A(start of work area)               (for the message to
DC  A(end of work area)                 be output on SYSLST)
DC  A(area in which the
      message to be edited              (V format for RTIO
      is stored)                        LL message for DCAM)
DC  A(control block generated
      with macro MDCBL) with device type
```

Return information in the control block when processing of input and output messages is error-free

Field ....EAL: Length of the required portion of main memory after processing of the input or output message.

Note

The length ascertained when an input message has been processed may be less than the length supplied after processing of an output message. The length of the printout should be that of the output message.

Field ....COL: Number of columns per line (device-dependent)

Note

If you wish to output on a printer the contents of the work area that, for example, you have saved in a file during the dialog, you have to take partial strings from the work area for each line to be printed. Each one of these partial strings must contain as many characters as are specified in the field ....COL. The work area contains no printer control characters.

The fields ....EAL and ....COL are only supplied with information if the user requests that the format should not be output on a printer by MFHSFORM.

10.2.3 Load MFHSFORR, MLINR macro

Operation	Operands
MLINR	{ name / (r) } [,prefix] [,mode]

Meaning of the operands:

name Name of a control block, up to 8 characters long, that was generated with the MDCBL macro.

(r) Decimal number or name, up to 8 characters long, of a register that must be loaded with the start address of a control block generated with the MDCBL macro

prefix Prefix up to 3 characters long.
If you do not specify a prefix, the default value MLR is assumed. MLINR generates the symbolic names prefix.00001, prefix.00002, prefix.00003,.... and RDT00001.

mode Specifies how the module MFHSFORR is to be loaded.

V If mode “V” is specified, MFHSFORR must be linked to the application program which then sets up the connection to MFHSFORR.
If no mode is entered, the module MFHSFORR is loaded by means of the MLINK macro. If the MFHSFORR module cannot be loaded, ...MRCF=X’0004’ and ....MSRC=X’1404’ are returned.
In either case the entry point is supplied in register 1; otherwise, register 1 cannot be used as the entry point.

The MLINR macro can only be used once in the program; any further MLINR calls are rejected. MLINR uses registers 1, 14 and 15. If necessary the user should save them before the macro call.

MLINR has XS capability.

10.2.4 Parameters and register entries

You can enter values in the registers either directly or via a parameter area.

Register entries, direct

Register	Contents
R13	Address of a save area, length of area: 18 full words
R14	Return address
R15	Entry point in the module MFHSFORR as supplied by the MLINR macro in register 1.
R1	Address of the FHSFORR parameter area
R0	Control information, length 4 bytes Byte 1-2 Pagination X‘0000‘ Internal pagination (1,2,3...) X‘nnnC‘ External pagination; the packed decimal number contained in these two bytes is used as the page number, e.g. X' 123C' for page 123. Byte 3: Specification of the access method X‘04‘ The message to be edited was generated for TIAM. X‘06‘ The message to be edited was generated for DCAM. Byte 4: Bit 2^0: Logging control =0 Header line is logged with internal and external pagination on every page. =1 No header line is logged. Bits 2^1 and 2^2 are reserved. Bit 2^3: Message type =0 Output message for data display terminal or printer =1 Input message from data display terminal Bit 2^4: Print control (for input messages only) =0 Work area is cleared prior to processing of the input message. =1 Work area is not cleared prior to processing of the input message. Bit 2^5: Output on a printer =0 Edited message is output on a printer. =1 Edited message is not output on a printer (for input and output messages). Bit 2^6: Switch for lowercase/uppercase letters =0 Lowercase letters are not converted =1 Output in uppercase letters Bit 2^7: Reserved

Note

The module MFHSFORR requires the work area in order to store the print-edited message. You should therefore insure that this area is sufficiently large, i.e. at least 2 Kbytes or 4 Kbytes for Unicode formats respectively.
No printer control characters are stored in the work area.
Data fields with the FHS attribute NOPRINT are not output on the printer. The print control specification “Do Not Clear Work Area” is only evaluated if positioning sequences are found in the input message, e.g. read mode = read modified. If any are found, the printedited format in the work area will not be deleted if bit 2^4 is set.
The data supplied by the data display terminal is incorporated into the format stored in the work area in accordance with the positioning sequences (field addresses). This enables a dialog step (input, output) on the terminal to be logged on the printer. Before the start of the next dialog step, i.e. prior to processing of the next message, the work area is always cleared.

Register entries, via a parameter area

```
Parameter area:
 DC  A(save area)                        for R13
 DC  A(return address)                   for R14
 DC  A(entry point in MFHSFORR)          for R15
 DC  A(0)  CONTROL INFORMATION           for R0
 DC  A(FHSFORR parameter area)           for R1

 FHSFORR parameter area:
 DC  A(start of work area)               (for the message to
 DC  A(end of work area)                 be output on SYSLST)
 DC  A(area in which the
       message to be edited              (V format for RTIO
       is stored)                        LL message for DCAM)
 DC  A(control block generated
       with macro MDCBL) with device type
 DC  A(second work area)
       internally used by MFHSFORR
```

Return information in the control block when processing of input and output messages is error-free

Field ....EAL: Length of the required portion of main memory after processing of the input or output message.

Note

The length ascertained when an input message has been processed may be less than the length supplied after processing of an output message. The length of the printout should be that of the output message.

Field ....COL: Number of columns per line (device-dependent)

Note

If you wish to output on a printer the contents of the work area that, for example, you have saved in a file during the dialog, you have to take partial strings from the work area for each line to be printed. Each one of these partial strings must contain as many characters as are specified in the field ....COL. The work area contains no printer control characters.

The fields ....EAL and ....COL are only supplied with information if the user requests that the format should not be output on a printer by MFHSFORR.

10.2.5 Return codes and error messages for MFHSFORM and MFHSFORR

....MRCF	....MSRC	Reg.15	Meaning
0000	0000	0000 0000	Error-free run
--	--	0000 0004	Register save area missing
--	--	0000 0008	Address of parameter block missing
--	--	0000 000C	Control block address missing
0010	1004	0000 0010	Address of message area missing
	1008		Start address of a work area missing
	100C		End address of a work area missing
	1010		End address of work area less than/equal to start address of work area
	1014		No device type specified
	1018		No access method specified
	101C		Page number not a decimal number
	1020		Device type not supported by FHS
	1024		Message length missing in message to be processed
	1028		Work area too small
	102C		No input data found
	1030		No output data found
	103C		Coded character set of format incompatible with current SYSLST assignment.

Note

If no return address is specified, the program is terminated with DUMP (TERM).

10.3 FHS code tables

Formats based on the character sets of other computers can be displayed (and printed) by using code tables. The designations for different character sets are contained in the IFG format.

In the case of 8-bit devices, the format contains information on the CCSN (Coded Character Set Name) to be used for formatting. See the "XHCS" User Guide for more details.

The CCS name is output in the user control block by FHS with the macro MCMAP. The CCS name for 7-bit formats contains blanks. For 8-bit formats, an addressable field of 8 bytes containing the designation "CSSN" is added to the control block (see also page 248ff).

A new service routine "2" (FHSSERV) with the parameter "FHS-CCSN-PAR" is offered in COBOL. This subroutine loads the format and stores the CCSN in the input field "FHS-CCSN-INFO" of the parameter list "FHS-CCSN-PAR".

User-specific code tables can also be used with MFHSCTAB.

If you have not defined any user-specific code tables, the character sets specified in the format will be used with an XHCS call. If no such character sets exist, the FHS default character sets are used.

If a format other than the FHS default character set is to be output and no XHCS exists, a corresponding return code is issued.

If different code tables exist and the specified table is not in the list of code tables available in the system, a corresponding return code is issued.

10.4 Creating the user-own code table module MFHSCTAB

The FHS code tables are supplied in the module MFHSCTAB. It is possible to use a user-own variant.

The set of tables comprises seven individual tables in the following order:

– Definition of the printable characters
– Conversion table from lowercase to uppercase letters for input fields (international)
– Conversion table from lowercase to uppercase letters for input fields (national)
– Checking table for relevant characters
– Checking table for alphabetic characters
– Checking table for arithmetic characters
– Conversion table for special characters

These tables are not independent of one another; note the following, for example: if code positions in the first three tables are redefined from the default assignment X’3F’ with other values, these code positions must be redefined as relevant in the fourth table.

10.4.1 Generating a user-own table set

Generation of the standard table set is performed by the macro MGCTS. This macro is contained in the macro library SYSLIB.FHS.082. When a user-own table macro is created, the default table set is modified by redefinition.

MGCTS macro

Operation	Operands
MGCTS	(000,macroname)

Meaning of the operands

000 Predefined parameter value, reserved for future development.

macroname Name of the user-specific redefinition macro, length not exceeding 7 characters.

User-own table macro:

```
            MACRO
&MACNAM     <macroname>              Prototype statement
            GBLC  &GVAR1
&MACNAM     EQU   *
            ORG   DEFTB&GVAR1.n+X'dist'
            DC    X'hexcode'
            .
            .
             MEND
```

where:

DEFTB&GVAR1.n (0 < n < 8) = Addresses of the seven default tables

macroname = Identical to macroname in the MGCTS call

Structure of the source for MFHSCTAB

```
MFHSCTAB   CSECT (optional attribute list)
MFHSCTAB   AMODE ANY              (optional)
MFHSCTAB   RMODE ANY              (optional)
           MCALL macroname
           MGCTS (000,macroname)
           END
```

10.4.2 Example of definition and generation of a user-own table module

```
            MACRO
&MACNAM     USERTABX
            GBLC  &GVAR1
*
*         LOWERCASE/UPPERCASE CONVERSION OF UMLAUTS
*         ON INPUT FROM INTERNATIONAL KEYBOARD
*
*
&MACNAM     EQU   *
            ORG   DEFTB&GVAR1.2+X'4F'
            DC    X'BC'
            ORG   DEFTB&GVAR1.2+X'FB'
            DC    X'BB'
            ORG   DEFTB&GVAR1.2+X'FD'
            DC    X'BD'
*
            MEND
```

User-own source for MFHSCTAB:

```
MFHSCTAB    CSECT PAGE
MFHSCTAB    AMODE ANY
MFHSCTAB    RMODE ANY
            MCALL USERTABX
            MGCTS (OOO,USERTABX)
            END
```

10.5 Use of XHCS tables

The tables of the **XHCS set** are contained in the GNLMTAB module of the library SYSSRC.XHCS.SYS.010.GNLMTAB. Unlike the user's own tables, an XHCS table lets you assign a different set of tables to each format. To find out how to generate and modify these tables refer to the "XHCS" User Guide.

An XHCS set of tables contains five tables:

– a table for defining the characters that can be represented
– a table for converting lowercase letters to upper case
– a table for permitted characters
– a table of alphabetic characters
– a table for special characters (not used).

Of the three sets of tables, the user's own set has the highest priority and the default set delivered with FHS the lowest. This means that if there is a user's own set, only this format is used for editing. The XHCS set can only be used if there is no user's own set. If there is neither a user's own nor an XHCS set, the default set is used.

Code name

To edit fields using an extended character set, you need the code name of this character set. This name is defined in IFG when a format is defined. Each code has its own name. In DCAM and TIAM applications it is possible to determine the name of the character set using FHSSERV; see page 372. This also applies to Fortran and PL/I.

11 Appendix

11.1 Examples of addressing aids

11.1.1 ASSEMBLER

Data transfer area not aligned, no attribute fields

Input formatting

```
         MACRO
         DELIVERI
*FORMAT  NAME: DELIVER
DELIVERI DS    0CL386
ADDRESSI DS    0C

NAMEI    DS    CL25
STREETI  DS    CL26
ZIPCODEI DS    CL5
CITYI    DS    CL24
ADDRESSG EQU *-ADDRESSI
CUSTMRNI DS    CL12
DATEI    DS    CL10
ARTICLEI DS    CL5
DESIGI   DS    CL28
QTYI     DS    CL6
UNITPRII DS    CL7
PRICEI   DS    CL14
         DS    CL5
         DS    CL28
         DS    CL6
         DS    CL7
         DS    CL14
         DS    CL5
         DS    CL28
         DS    CL6
         DS    CL7
```

```
          DS    CL14
          DS    CL5
          DS    CL28
          DS    CL6
          DS    CL7
          DS    CL14
SUBTOTAI  DS    CL15

SALESTAI DS    CL14
TOTALI   DS    CL15
         MEND
```

Output formatting

```
          MACRO
          DELIVERO
*FORMAT   NAME: DELIVER
DELIVERO  DS    0CL386
ADDRESSO  DS    0C
NAMEO     DS    CL25
STREETO   DS    CL26
ZIPCODEO  DS    CL5
CITYO     DS    CL24
ADDRESSF  EQU *-ADDRESSO
CUSTMRNO  DS    CL12
DATEO     DS    CL10
ARTICLEO  DS    CL5
DESIGO    DS    CL28
QTYO      DS    CL6
UNITPRIO  DS    CL7
PRICEO    DS    CL14
          DS    CL5
          DS    CL28
          DS    CL6
          DS    CL7
          DS    CL14
          DS    CL5
          DS    CL28
          DS    CL6
          DS    CL7
          DS    CL14
          DS    CL5
          DS    CL28
          DS    CL6
          DS    CL7
          DS    CL14
SUBTOTAO  DS    CL15
```

```
SALESTAO DS    CL14
TOTALO   DS    CL15
         MEND
```

Data transfer area with separate attribute blocks and field contents

```
*
         DELIVER
*FORMAT  NAME: DELIVER
*
EUAGA    DSECT
GARCMAIN DS    F           RC MAIN
GARCCTGR DS    H           RC CATEGORY
GARCREAS DS    H           RC REASON
GAFLDMOD DS    CL1         FIELDS MOD
GAFLDDET DS    CL1         FIELDS DET
GAFLDVAL DS    CL1         FIELDS VALID
GAUSEXRC DS    CL1         USER EXIT RC
GAFLDUND DS    CL1         FIELDS UNDEFINED
GAIKEYCL DS    CL1         INPUT KEY CLASS
GAIKEYNB DS    H           INPUT KEY NUMBER
         DS    CL4         RESERVED
GAINTCTL DS    CL1         INIT CTL
GAINTOPT DS    CL1         INIT OPT
GATABCTL DS    CL1         TAB CTL
GAFCTLCK DS    CL1         FCT LOCK
GAVMICTL DS    CL1         VMI CTL
GAHMICTL DS    CL1         HMI CTL
         DS    CL2         RESERVED
GACYCCTL DS    CL1         CYCLE CTL
GACOPCTL DS    CL1         COPY CTL
GAARMCTL DS    CL1         ALARM CTL
GAHOLECO DS    CL1         HOLE COLOR
GADISSEL DS    CL1         DISPLAY SEL
GALEVSEL DS    CL1         LEVEL SEL
GAOUTMOD DS    CL1         OUTPUT MODE
GACURCTL DS    CL1         CURSOR CTL
GACURPOS DS    F           CURSOR POS
GAUSEXCT DS    CL1         USER EXIT CTL
         DS    CL1         RESERVED
GASTARTL DS    H           STARTLINE
GAPKEYST DS    CL8         P KEY SET
EUAGAL   EQU   *-EUAGA
*
*
*
DELIVERB DSECT
```

```
DELIVERS DS    0CL4       BASIC ATTR
DELIVERI DS    CL1          INPUT STATE
DELIVERT DS    CL1          INPUT STATE ACT
DELIVERE DS    CL1          EDIT STATE
DELIVERO DS    CL1          OUTPUT CTL
DELIVERL EQU   *-DELIVERB
*
DTP      CSECT
*
         DS    0F
DELIVER  DS    0CL00528
DELIVERG DS    CL(EUAGAL)
*
*
DELIVERA DS    0CL00117
NAMEA    DS    CL(DELIVERL)
STREETA  DS    CL(DELIVERL)
ZIPCODEA DS    CL(DELIVERL)
CITYA    DS    CL(DELIVERL)
CUSTMRNA DS    CL(DELIVERL)
DATEA    DS    CL(DELIVERL)
ARTICLEA DS    CL(DELIVERL)
DESIGA   DS    CL(DELIVERL)
QTYA     DS    CL(DELIVERL)
UNITPRIA DS    CL(DELIVERL)
PRICEA   DS    CL(DELIVERL)
         DS    CL(DELIVERL)
         DS    CL(DELIVERL)
         .
         .
         .
         DS    CL(DELIVERL)
         DS    CL(DELIVERL)
SUBTOTAA DS    CL(DELIVERL)
SALESTAA DS    CL(DELIVERL)
TOTALA   DS    CL(DELIVERL)
*
*
DELIVERD DS    0CL00361
ADDRESSD DS    0C
NAMED    DS    CL025
STREETD  DS    CL026
ZIPCODED DS    CL005
CITYD    DS    CL024
ADDRESSG EQU   *-ADDRESSD
CUSTMRND DS    CL012
DATED    DS    CL014
ARTICLED DS    CL005
```

```
DESIGD   DS    CL028
QTYD     DS    CL005
UNITPRID DS    CL006
PRICED   DS    CL011
         DS    CL005
         DS    CL028
         .
         .
         .
         DS    CL006
         DS    CL011
SUBTOTAD DS    CL012
SALESTAD DS    CL011
TOTALD   DS    CL012
*
          .
          .
          .
          .
```

11.1.2 COBOL

Data transfer area not aligned, no attribute fields

Input formatting

```
* IFG COPY   NAME: EDELIVER RELATED COPY NAME: ADELIVER
*     FORMAT NAME: DELIVER  USER AREA LENGTH : 00386   UTM-TYPE: *
     41  ADDRESSI.
       42  NAMEI                          PIC X(025).
       42  STREETI                        PIC X(026).
       42  ZIPCODEI                       PIC 9(005).
       42  CITYI                          PIC X(024).
     41  CUSTMRNOI                      PIC 9(012).
     41  DATEI                          PIC X(010).
     41  ARTLINE-TABI.
       42  ARTLINEI                        OCCURS 04 TIMES.
         43    ARTICLENUMBERI                 PIC 9(005).
         43    DESIGNATIONI                   PIC X(028).
         43    QUANTITYI                      PIC 9(006).
         43    UNITPRICEI                     PIC 9(007).
         43    PRICEI                         PIC X(014).
     41  SUBTOTALI                      PIC X(015).
     41  SALESTAXI                      PIC X(014).
     41  TOTALI                         PIC X(015).
```

Output formatting

```
* IFG COPY   NAME: ADELIVER RELATED CPOY NAME: EDELIVER
*     FORMAT NAME: DELIVER  USER AREA LENGTH : 00386   UTM-TYPE: *
     41  ADDRESSO.
       42  NAMEO                          PIC X(025).
       42  STREETO                        PIC X(026).
       42  ZIPCODEO                       PIC 9(005).
       42  CITYO                          PIC X(024).
     41  CUSTMRNOO                      PIC 9(012).
     41  DATEO                          PIC X(010).
     41  ARTLINE-TABO.
       42  ARTLINEO                        OCCURS 04 TIMES.
         43    ARTICLENUMBERO                 PIC 9(005).
         43    DESIGNATIONO                   PIC X(028).
         43    QUANTITYO                      PIC 9(006).
         43    UNITPRICEO                     PIC 9(007).
         43    PRICEO                         PIC X(014).
     41  SUBTOTALO                      PIC X(015).
     41  SALESTAXO                      PIC X(014).
     41  TOTALO                         PIC X(015).
```

Data transfer area with separate attribute blocks and field contents

```
* IFG COPY   NAME: DELIVER
*     FORMAT NAME: DELIVER  USER AREA LENGTH : 00528   UTM TYPE: #

************************************************************
*                 GLOBAL ATTRIBUTE BLOCK                   *
************************************************************

    40 DELIVER-GLOBALS.
*        FORM-RETURNCODE
      41 RC-MAIN                      PIC 9(5) COMP SYNC.
      41 RC-CATEGORY                  PIC 9(4) COMP.
      41 RC-REASON                    PIC 9(4) COMP.
*        FORM-INDICATORS
      41 FIELDS-MOD                   PIC X.
      41 FIELDS-DET                   PIC X.
      41 FIELDS-VALID                 PIC X.
      41 USER-EXIT-RC                 PIC X.
      41 FIELDS-UNDEFINED             PIC X.
*        INPUT-IDENTIFICATION
      41 INPUT-KEY-CLASS              PIC X.
      41 INPUT-KEY-NUMBER             PIC 9(4) COMP.
      41 FILLER                       PIC X(4).
*        DEVICE-CONTROLS
      41 INIT-CTL                     PIC X.
      41 INIT-OPT                     PIC X.
      41 TAB-CTL                      PIC X.
      41 FCT-LOCK                     PIC X.
      41 VMI-CTL                      PIC X.
      41 HMI-CTL                      PIC X.
      41 FILLER                       PIC X(2).
*        OUTPUT-CONTROLS
      41 CYCLE-CTL                    PIC X.
      41 COPY-CTL                     PIC X.
      41 ALARM-CTL                    PIC X.
      41 HOLE-COLOR                   PIC X.
*        FORM-CONTROLS
      41 DISPLAY-SEL                  PIC X.
      41 LEVEL-SEL                    PIC X.
      41 OUTPUT-MODE                  PIC X.
      41 CURSOR-CTL                   PIC X.
      41 CURSOR-POS                   PIC 9(5) COMP.
      41 USER-EXIT-CTL                PIC X.
      41 FILLER                       PIC X.
      41 STARTLINE                    PIC 9(4).
*
      41 P-KEY-SET                    PIC X(8).
```

```
*************************************************************
*                  FIELD ATTRIBUTE BLOCKS                   *
*************************************************************

    40 DELIVER-ATTR.
      41 ADDRESS-FAB.
        42 NAME-FAB.
          43 BASIC-ATTR.
            44 INPUT-STATE              PIC X.
            44 INPUT-STATE-ACT          PIC X.
            44 EDIT-STATE               PIC X.
            44 OUTPUT-CTL               PIC X.

        42 STREET-FAB.
          43 BASIC-ATTR.
            44 INPUT-STATE              PIC X.
            44 INPUT-STATE-ACT          PIC X.
            44 EDIT-STATE               PIC X.
            44 OUTPUT-CTL               PIC X.

        42 ZIPCODE-FAB.
          43 BASIC-ATTR.
            44 INPUT-STATE              PIC X.
            44 INPUT-STATE-ACT          PIC X.
            44 EDIT-STATE               PIC X.
            44 OUTPUT-CTL               PIC X.

        42 CITY-FAB.
          43 BASIC-ATTR.
            44 INPUT-STATE              PIC X.
            44 INPUT-STATE-ACT          PIC X.
            44 EDIT-STATE               PIC X.
            44 OUTPUT-CTL               PIC X.

      41 CUSTMRNO-FAB.
        42 BASIC-ATTR.
          43 INPUT-STATE              PIC X.
          43 INPUT-STATE-ACT          PIC X.
          43 EDIT-STATE               PIC X.
          43 OUTPUT-CTL               PIC X.

      41 DATE-FAB.
        42 BASIC-ATTR.
          43 INPUT-STATE              PIC X.
          43 INPUT-STATE-ACT          PIC X.
          43 EDIT-STATE               PIC X.
          43 OUTPUT-CTL               PIC X.
```

```
41 ARTLINE-TAB-FAB.
  42 ARTLINE-FAB                      OCCURS 04 TIMES.
    43 ARTICLENUMBER-FAB.
      44 BASIC-ATTR.
        45 INPUT-STATE                PIC X.
        45 INPUT-STATE-ACT            PIC X.
        45 EDIT-STATE                 PIC X.
        45 OUTPUT-CTL                 PIC X.

    43 DESIGNATION-FAB.
      44 BASIC-ATTR.
        45 INPUT-STATE                PIC X.
        45 INPUT-STATE-ACT            PIC X.
        45 EDIT-STATE                 PIC X.
        45 OUTPUT-CTL                 PIC X.

    43 QUANTITY-FAB.
      44 BASIC-ATTR.
        45 INPUT-STATE                PIC X.
        45 INPUT-STATE-ACT            PIC X.
        45 EDIT-STATE                 PIC X.
        45 OUTPUT-CTL                 PIC X.

    43 UNITPRICE-FAB.
      44 BASIC-ATTR.
        45 INPUT-STATE                PIC X.
        45 INPUT-STATE-ACT            PIC X.
        45 EDIT-STATE                 PIC X.
        45 OUTPUT-CTL                 PIC X.

    43 PRICE-FAB.
      44 BASIC-ATTR.
        45 INPUT-STATE                PIC X.
        45 INPUT-STATE-ACT            PIC X.
        45 EDIT-STATE                 PIC X.
        45 OUTPUT-CTL                 PIC X.

41 SUBTOTAL-FAB.
  42 BASIC-ATTR.
    43 INPUT-STATE                  PIC X.
    43 INPUT-STATE-ACT              PIC X.
    43 EDIT-STATE                   PIC X.
    43 OUTPUT-CTL                   PIC X.

41 SALESTAX-FAB.
  42 BASIC-ATTR.
    43 INPUT-STATE                  PIC X.
    43 INPUT-STATE-ACT              PIC X.
```

```
          43 EDIT-STATE               PIC X.
          43 OUTPUT-CTL               PIC X.

      41 TOTAL-FAB.
        42 BASIC-ATTR.
          43 INPUT-STATE              PIC X.
          43 INPUT-STATE-ACT          PIC X.
          43 EDIT-STATE               PIC X.
          43 OUTPUT-CTL               PIC X.

    40 DELIVER-ATTR-TAB REDEFINES DELIVER-ATTR OCCURS 029 TIMES.
      41 BASIC-ATTR.
        42 INPUT-STATE              PIC X.
        42 INPUT-STATE-ACT          PIC X.
        42 EDIT-STATE               PIC X.
        42 OUTPUT-CTL               PIC X.

************************************************************
*                    FIELD DATA PART                       *
************************************************************

    40 DELIVER-DATA SIGN IS TRAILING SEPARATE.
      41 ADDRESS.
        42 NAME                           PIC X(025).
        42 STREET                         PIC X(026).
        42 ZIPCODE                        PIC X(005).
        42 CITY                           PIC X(024).
      41 CUSTMRNO                       PIC X(012).
      41 DATE                           PIC X(014).
      41 ARTLINE-TAB.
        42 ARTLINE                        OCCURS 04 TIMES.
          43 ARTICLENUMBER                  PIC X(005).
          43 DESIGNATION                    PIC X(028).
          43 QUANTITY                       PIC  9(005).
          43 UNITPRICE                      PIC S9(003)V9(002).
          43 PRICE                          PIC S9(008)V9(002).
      41 SUBTOTAL                       PIC S9(009)V9(002).
      41 SALESTAX                       PIC S9(008)V9(002).
      41 TOTAL                          PIC S9(009)V9(002).
```

11.2 Return codes

11.2.1 Return codes in ASSEMBLER programs

The return codes in fields ....MRCF and ....MSRC provide information about the execution of the MOMAP, MCMAP and MULIB macros.

If execution is error-free, both fields will contain X'0000'.

The return code in field ....MRCF is also contained in the two low-order bytes of register 15.

If the return code could not be stored in the control block because the control block is not addressable, the return code X'000C' will be entered in register 15.
In such a case the user must check that no part of the control block lies within the code of the MGMAP macro.

Return codes after execution of the MOMAP macro

....MRCF	Meaning of return code	
0000	Formatting opened without errors	
0004	Error during opening	The secondary return code in field ....MSRC indicates the type of error

Secondary return code after execution of the MOMAP macro

....MSRC	Meaning	Error recovery action
0000	Formatting opened without errors	
0404	Loaded format definition(s) not correct	Correct format definition or redefine format
0804	Format definition(s) cannot be loaded	Validate the format names in RESMAP operand of MGMAP macro; check if the formats are in the format application file
1404	The formatting routine cannot be loaded	Check that formatting routine is available in the specified FHS module library
1408	Loaded formatting routine is not correct	
140C	Formatting routine and MGMAP macro are inconsistent	Check the version number of formatting routine (MFHSROUT); it must be equal to or greater than the version number of the MGMAP macro used

Return codes after execution of the MCMAP macro

....MRCF	Meaning of the return code	
0000	Input or output formatting completed without errors	
0004	Error during formatting	The secondary return code in the field ....MSRC indicates the type of error
0008	Formatting performed with automatic error correction	The secondary return code in the field ....MSRC is either X' 0000' (no further effect) or indicates the corrected error
0010	Error caused by the terminal user.	The secondary return code in the field ....MSRC specifies the type of error

Secondary return code after execution of the MCMAP macro

....MSRC	Meaning	Error recovery action
0004	Value given in the MAPCNT operand of the MGMAP macro is less than the number of formats used. Formats are reloaded at each call since no entry is made in the directory	Increase value in MAPCNT operand
000C	A restart address other than 0 was specified for a printer terminal. The restart function is not executed	
04XX	Format incorrect	Validate format name in MCMAP macro; check if the format is stored under the correct name in the format application file
0404	See corresponding return code for MOMAP; or for ....MRCF=X' 0010' : wrong control character sequence in the input message, e.g. AM	Call MOMAP before MCMAP; enter only valid characters on the terminal
0408	Subformat loaded instead of main format; or for ....MRCF=X' 0010' : format on the screen was destroyed	Validate MCMAP macro e.g. terminal was switched off For the 3270: the CLEAR key was pressed or Attention Field type 1 was selected
040C	Format was not generated with FHS macros or if ....MRCF=X' 0010' : input mode not identifiable; Possible causes: – A format for fast formatting that was not generated for data terminal type 8160/97xx was output on a terminal of this type and the DÜ2 key pressed. – A format for normal formatting was output on a type 97xx terminal and the DÜ2 key pressed. The format was not generated for this data display terminal and 97xx was not passed as the device type. For the 3270: format error	Retranslate format Define the format for the correct terminal type or do not use DÜ2 key Define the format for the correct terminal type or specify the correct terminal type in MDCBL or do not use the DÜ2 key

....MSRC	Meaning	Error recovery action
0410	Neither input nor output formatting specified, or if ....MRCF=X' 0010' : no automatic correction for PROTRET fields because there is no restart area. For the 3270: A field that should be included in the input message is missing. Automatic correction is performed, i.e. – missing PROTRET fields are supplied with the associated field contents from the restart area. If there is n restart area, no automatic correction takes place; – missing UNPROT fields (in ISTD=RUNP mode) are regarded as deleted and filled with input-fill or NULL characters	Specify in the MCMAP macro whether input or output formatting is required
0414	Format is not decodable because it was not generated with FHS macros For the 3270: illegal field type (format error)	Retranslate format
0418	Format loaded instead of subformat	Validate subformat name in the format definition
041C	Format not generated with FHS macros	Retranslate format
0420	Defined format not correct	Call MOMAP before MCMAP; check that defined formats have not been overwritten
0424	Format to be loaded not correct	Check format application file
0428	XHCS is not available	Install XCHS (available as of BS2000 V10).
08XX	Format not loadable	Validate format name in MCMAP macro; check format application file
0804	See corresponding return code for MOMAP	
0808	Non-loadable format called	Call MOMAP before MCMAP; if call sequence is correct, check MOMAP return code
080C	Format to be loaded is not loadable	Validate MCMAP call or check available memory
0C04	Call to third-level subformat (see page 560)	Validate format and subformat definitions
14XX	Formatting routine not correct or not loadable	Check that formatting routine MFHSROUT is available in the specified FHS module library

....MSRC	Meaning	Error recovery action
1404 1408 140C	See corresponding return code for MOMAP macro	
1410	Formatting routine not correct or not loadable despite several attempts	Check return code after MOMAP call
1424	Formatting routine module cannot be loaded	Check whether all the modules of the formatting routine are available in the library
20XX	Formatting error	Validate format definition
2004	Transfer area and/or restart area do(es) not begin on a halfword boundary	Align start address of transfer area and/or restart area on a halfword boundary
2008	Terminal input/output area is located within MGMAP area	Define terminal input/output area outside MGMAP area
200C	Output area or input area is too small	Increase value of IOLEN operand in MGMAP macro call
2010	Too many data fields per line specified for 975x Data Display Terminals	Correct format definitions
2014	Format specified in global attribute 'P-Key Set' is not a P-key format	Alter entry in global attribute 'P-Key Set'
2018	P-key format was generated with control statement MSG	Generate P-key format without using control statement MSG
201C	Data field too long; screen overflow	Check format definition
2020	Transfer area and/or restart area is located within MGMAP area; transfer area too small or start address incorrect; value in UARLEN= operand in MDCBL or MUCBL greater than transfer area in program	Define transfer area and/or restart area outside MGMAP area; increase size of transfer area; correct value in UARLEN operand
2024	Input message not decodable; possible reasons: – field address in the input message invalid – field address in the input message missing – field length in the input message too large	Do not enter any illegal ESC sequences or invalid control characters
2028	Absolute or relative line entry too large (causes screen overflow or shifts format over the last line on the screen)	Validate positioning specifications in the format definition
202C	Absolute column entry too large (line overflow)	-

....MSRC	Meaning	Error recovery action
2030	Absolute line or column specification causes backward positioning.	Validate positioning specifications in the format definition
2034	Value of UARLEN operand in MDCBL or MUCBL too small	Increase value of UARLEN operand or alter format definition
2038	Invalid specification of terminal type by TSTAT or TMODE macro	Correct entry in MUCBL or use correct terminal
203C	Formats using all the functions of a data display terminal can not be output on data display terminals having a more restricted functional scope For the 3270: invalid device type	Use correct terminal or ' select' correct line
2040	Specified modification of number of characters per line not possible for the format	Use correct terminal; select correct line
2044	Maximum number of columns excee-ded, for printer output only. The number of possible printing positions is dependent upon the printer type and the entries in the control block for HMI and PAPER	Check the format and the entries in the MDCBL for HMI and PAPER
2048	Device error on terminal; 0010 is returned in MRCF	Inform system diagnostics staff
2050	Format not suitable for the terminal; 0010 is returned in MRCF	Generate format correctly
2054	HMI was changed in macro MDCBL or MUCBL for formats with 'fast formatting' and PCL printer	Either do not change the macro or use a format without ' fast formatting'
2058	The coded character set is not available for 8-bit formats.	Incorporate the coded character set into XHCS.
24XX	Error during partial formatting	
2404	MSTD=RSET or RSON required for a partial format not displayed on screen	First output format with MSTD=BEGN or ONLY
2410	Restart area too small	Increase restart area
2414	Address of restart area not found in MAPLIST, output formatting without a restart followed by input formatting with a restart	Check application program; request restart for output formatting as well
2418	Partial formatting requested and MAPPART operand not specified	Define MAPPART operand in MDCBL or MUCBL macro accordingly

....MSRC	Meaning	Error recovery action
241C	CLEAR=yes was specified during an open partial formatting cycle (output formatting)	Only specify CLEAR=NO during a partial formatting cycle
2420	For opening a partial formatting cycle (during output formatting) CLEAR=NO was specified	Specify CLEAR=yes to open a partial formatting cycle
2424	Message input/output area changed during partial formatting cycle (with output formatting)	Check application program
2428	Specified format not generated with IFG	Generate partial formats only with IFG
242C	Specified device type not supported for partial formatting	Use correct terminal or do not use partial formatting
2430	No length specified for restart area	Enter length of restart area in MAPLIST area
2434	A line to be formatted was already formatted (two partial formats overlap on the screen)	Check format definition; if necessary redefine number of start line
2438	Number of start line changed	Check start line number
243C	Type of device changed during a partial formatting cycle	Device type must not be changed during a partial formatting cycle
2440	Input mode changed during a partial formatting cycle	Input mode must not be changed during a partial formatting cycle (MDCBL operands ISTD or NILS)
2444	Various types of devices used for input and output formatting	Device type must not be changed between input and output formatting
2448	MDCBL operand PMOD=yes can no longer be executed	Do not change the value of the MDCBL operand PMOD during a partial formatting cycle. Formatting performed as specified in the first partial formatting call of the cycle
244C	MDCBL operand PMOD ignored	
2450	Partial formatting cycle for output not terminated	Terminate partial formatting cycle with MAPPART=LAST
2454	Message input area changed during a partial formatting cycle	Use one input area throughout a partial formatting cycle
2458	Partial format unknown during input formatting	Repeat partial formatting cycle for output
245C	Number of start line changed during input formatting	Check start line number
2460	Input formatting already exists for this partial format	None; partial format is formatted again

....MSRC	Meaning	Error recovery action
2464	No partial format contains input data from the terminal	None; nothing was formatted
2468	Partial format is formatted but MAPLIST operand not specified	The partial format is formatted normally taking account of the start line number
246C	Partial formatting not in MAPLIST area (serial input formatting)	Repeat output formatting cycle
2470	RESTART operand changed in the current output cycle; for #formats: screen generation was started without the restart function with partial formats having the old structure of the data transfer area	Maintenance of the restart area is mandatory with #formats; avoid mixed operation as far as possible
2474	Execution of restart was required before restart areas were created	Require restart only if a restart area is being used during formatting as well
2478	Execution of restart was required but no format name located in the MAPLIST area (MAPLIST area destroyed)	Check MAPLIST area
247C	Execution of restart was required but no address of the restart area located	Check if RESTART=ye was specified before RESTART=EX
2480	MCMAP operand MAPLIST was specified; however, the format to be output is not a partial format and MAPPART=LAST was not specified	Check format name; if valid, output format again with MAPPART=LAST
2484	MCMAP operand MAPLIST was specified; however, the format is not a partial format	The format is formatted normally
2488	Different structure of the MAPLIST and control block transfer areas (when updating a partial format)	Check format definition
248C	MAPLIST area of old structure and screen dimension not 24 x 80	Create new MAPLIST area using FHS V6 or later
2490	Invalid screen dimension	Check format
2494	Screen dimension was changed in the current output cycle	Use only partial formats having the same screen dimension in the output cycle
249C	Invalid character in Unicode message	Check that the format was output as a Unicode message by VTSU (via MODIFY-TERMINAL-OPTIONS or VTSU control block)
24A0	BS2000 version not compatible with Unicode format.	You need at least BS2000/OSD V06.0B.
24A4	Unicode characters have been entered in a not-UNICODE field.	Re-output the format with an error message

....MSRC	Meaning	Error recovery action
24A8	Unicode partial format not in Unicode cycle	Check the consistence of partial formats (Unicode property)
2804	Additional memory cannot be requested	Restart program
30XX	Error in input/output area during input formatting	Check program
3004	Input message not decodable. For the 3270: AID byte is missing or invalid	Validate address of input/ output area; was the access method specified correctly in MGMAP?
3008	CDS (“code-transmit-key”) byte cannot be decoded	Check if the input/output area was overwritten before the MCMAP call
300C	Format not generated with IFG (format error)	
3010	Formats with ' fast formatting'cannot be output on a 3287 printer	Generate format without ' fast formatting'
3404	Incorrect terminal type	
3408	Terminal type in the format is 3287 and the value for CNTRLU is incorrect	Enter the correct value for CNTRLU for this terminal
340C	Terminal type in control block is 3287 and the value for CNTRLU in the control block is incorrect	Enter the correct value for CNTRLU for this terminal in the control block
3410	Format is longer than one page. A page may have up to 74 lines at 6 lines per inch or 98 lines at 8 lines per inch	Change format specification or reduce the line spacing
3414	Field length causes a page overflow	Check the format
3800	#format could not be formatted	See detailed return code in the control block
3804	No restart area for #formats for formatting on the data display terminal	Maintain restart area
3808	Errored field contents in the data transfer area on output formatting (only for fields with edit function)	Check data source, e.g. data base
380C	Undefined value for a global attribute or invalid combination of global attributes on output formatting	Check global attribute in data transfer area
3810	Undefined value for a field attribute or invalid combination of field attributes on output formatting	Check field attribute in data transfer area
3814	Formatting of a #partial format for data display terminal with no MAPLIST area	Format #partial format with partial formatting

....MSRC	Meaning	Error recovery action
3818	No terminal characteristics for #format	Before the MCMAP call the control block must be updated with MUCBL DEVAR=
381C	#format not suitable for variable positioning	Observe restrictions
3C04	MCMAP service call cannot be executed; possible causes: service function entry missing in control block or contains an undefined value	Check call
3C08	MCMAP service call 'Initialization of the Data Transfer Area' cannot be executed because the specified format is not an #format	#formats are prerequisite for this function
6000	Unrecognized control character sequence	Check that message has been transferred to data display terminal in its entirety
6004	Control character sequence for setting the screen dimension cannot be decoded	Check that message has been transferred to data display terminal in its entirety
6008	The screen dimension specified in the format is not supported by the data display terminal. FHS was unable to detect this as no device status data was transferred (see MDMEM)	Regenerate format, specifying different screen dimension
6020	Control character sequence for field display (field attributes) cannot be decoded	Check that message has been transferred to data display terminal in its entirety
6028/ 6034	Characters in a data field can not be displayed using the character set specified in the field attribute; the specified character set was not loaded previously. FHS was unable to detect this as no administrative area was transferred (see MDMEM)	Check format and regenerate if necessary
602C	Control character sequence for attribute modification cannot be decoded	Check that message has been transferred to data display terminal in its entirety
6038/ 6058/ 6060	Error message from the 9763 Data Display Terminal	Check that message has been transferred to data display terminal in its entirety
6050	Control character sequence for loading character sets cannot be decoded	Check that message has been transferred to data display terminal in its entirety
6054	The character set specified in the format cannot be loaded. The required character generator is not present in the device. FHS was unable to detect this as no device status data was transferred (see MDMEM)	Regenerate format, regenerate character set

....MSRC	Meaning	Error recovery action
605C	The character generator required for the caracter set to be loaded is in use by the data display terminal and not available. FHS was unable to detect this as no device status data was transferred (see MDMEM)	Regenerate format, regenerate character set file or change assignment of character generators
6064	The screen dimension specified in the format is not supported by the 9763 Data Display Terminal	Regenerate format, specifying a different screen dimension
6068	9763 is not specified as the device type in a character set file	Check character set file and regenerate if necessary
606C	Neither S (single plane) nor T (triple plane) is specified as the character set type in a character set file	Check character set file and regenerate if necessary
6070	The area for the output message is too small, not all the character sets specified in the format can be loaded	Extend message output area; regenerate format and reduce number of character sets to be loaded
6074	The characters in a data field cannot be displayed using the character set specified in the field attribute; the specified character set was not loaded previously.	Check format and regenerate if necessary
6078	A character set format, a P key format or a DE format was loaded instead of a format or a partial format.	Check format library and format name
607C	Character set file cannot be decoded	Check character set file and regenerate is necessary
6080	The character set specified in the format cannot be loaded. The required character generator is not present in the device	Regenerate format, regenerate character set
6084	The character generator required for the character set to be loaded is in use by the data display terminal and not available.	Regenerate format, regenerate character set file or change assignment of character generators

Note

The return codes for the MOMAP macro may also occur with the MCMAP macro if the MOMAP macro was not called explicitly.

11.2.2 Return code in COBOL programs

The return codes in the items FHS-MAIN-RC, ERROR-CATEGORY and ERROR-REASON provide details of the execution of the FHS COBOL calls and formatting.

FHS-MAIN-RC contains the primary return code. The value 0 indicates error-free execution of the FHS call and of formatting.

FHS-ERROR-INFO communicates the nature of the error (error category and reason are transferred in **ERROR-CATEGORY** and **ERROR-REASON** respectively).

Return codes after formatting

After a formatting run (i.e. after any of the DCAM/TIAM calls for input/output), the following return codes may occur:

Primary return code

FHS-MAIN-RC	Meaning
0	Input and/or output formatting was error-free.
4	Formatting error. Refer to secondary return code in FHS-ERROR-INFO for details.
8	Formatting was performed with automatic error handling. The secondary return code in FHS-ERROR-INFO is either 0 (no further effects) or gives the reason for the error correction.
16	The error was caused by the terminal user. The secondary return code in FHS-ERROR-INFO indicates the type of error.
24	A partial format was formatted with FHS-MAP-PART=“S” or “N”. FHS only formats; no input or output takes place (warning).

Secondary return code

ERROR-CATEGORY	ERROR-REASON	Meaning	Action for error recovery
0	4	More formats were used than specified in FHS-MAP-NO. Automatic error handling: formats are reloaded for each call since no entry in the directory can be made	
0	8	The partial format was only formatted and not output; the partial formatting cycle has not yet been terminated	Continue partial formatting cycle
0	12	The partial format was not formatted (input formatting). No input will take place	
4	4	Loaded format definition(s) is (are) incorrect, or, if FHS-MAIN-RC=16:incorrect control string in the input message e.g. AM	Correct or regenerate format definition(s), only enter valid characters at the terminal
4	8	A subformat was loaded instead of a format or, if FHS-MAIN-RC=16:the format on the screen has been destroyed.	Check format name e.g. terminal was switched off. For the 3270: the CLEAR key was pressed or Attention Field type 1 was selected
4	12	Format with invalid version number or for FHS-MAIN-RC=16: Input mode not recognized; Possible causes: – A format for fast formatting, that was not generated for the 97xx terminal type, was output on a terminal of this type and the DÜ2 key was pressed. – A format for normal formatting was output on a 97xx terminal type and the DÜ2 key was pressed. The format was not generated for this data display terminal type and 97xx was not transferred as device type. For the 3270: format error	Check format definition Define format for correct terminal type or do not use DÜ2 key Define format for the correct terminal type, specify correct terminal type or do not use DÜ2 key.

ERROR-CATEGORY	ERROR-REASON	Meaning	Action for error recovery
4	16	For the 3270: A field that should be included in the input message is missing. No automatic correction is performed, i.e. – missing PROTRET fields are supplied with the associated field contents from the restart area. If there is no restart area, no automatic correction takes place; – missing UNPROT fields (in ISTD=RUNP mode) are regarded as deleted and filled with input-fill or NULL characters	
4	20	Format cannot be used. For the 3270: illegal FDB field type (format error)	Correct or regenerate format definition
4	24	A format was loaded instead of a subformat	Check subformat name in format definition
4	28	Subformat with invalid version number	Retranslate subformat
4	32	Loaded format is incorrect	Correct or regenerate format definition
4	36	Format to be reloaded is incorrect	Check/correct format definition
4	40	XHCS is not available	Install XHCS (available as of BS2000 V10)
8	4	One or more formats cannot be loaded on opening formatting	Check format names and whether the formats are present in the format application file
8	8	Format not loadable	First call FHSINIT; f correct, check return code from FHSINIT
8	12	Format to be reloaded is not loadable	Check format name or available memory
12	4	Third-level subformat was called see page 560	Check subformat definition
20	4	Formatting routine MFHSROUT cannot be loaded	Check whether MFHSROUT is present in the specified FHS module library
20	8	Loaded formatting routine MFHSROUT is incorrect	

ERROR-CATEGORY	ERROR-REASON	Meaning	Action for error recovery
20	12	Loaded formatting routine MFHSROUT is incompatible with FHS version	Check version number of FHS and MFHSROUT
20	16	Formatting routine module cannot be loaded	Check whether all modules of the formatting routine are present in the library
20	36	Formatting routine module cannot be reloaded	Check whether all modules in the library of the formatting routine are present
32	4	Transfer area not aligned on halfword boundary	Align start address of transfer area on halfword boundary
32	16	Too many data fields per line specified for 975x Data Display Terminals	Correct format definition
32	20	Format specified in global attribute 'P-Key Set' is not P-key format	Alter entry in global attribute 'P-Key Set'
32	24	P-key format generated with control statement MSG	Generate P-key format without using MSG
32	28	Data field too long, screen overflow	Check format definition
32	32	Invalid address or length of input transfer area	Check/extend input transfer area
32	36	Input message not decodable; possible reasons: – field address in the input message invalid – field address in the input message missing – field length in the input message too large	Do not enter any illegal ESC sequences or invalid control characters
32	40	Absolute or relative line entry too large (causes screen overflow or shifts format over the last line on the screen)	Check position entries in format definition
32	44	Absolute column specification too large, screen overflow	
32	48	Absolute line or column specification overwrites previously defined field (reverse positioning).	Check attribute fields
32	52	MAP-USER-AREA-LEN in FHS-INIT-PAR too small	Specify greater value for MAP-USER-AREA-LEN
32	56	Invalid terminal type was specified	Correct entry or use appropriate terminal

ERROR-CATEGORY	ERROR-REASON	Meaning	Action for error recovery
32	60	Formats using all the functions functions of a data display terminal cannot be output on Data Display Terminals having a more restricted functional scope For the 3270: invalid device type	Use correct terminal
32	68	Maximum number of columns exceeded; printer outputs only. The possible number of printing positions is dependent on the printer type and the entries in MAP-PRINT-COLUMNS and MAP-PRINT-PAPER	Check format and entries in MAP-PRINT-COLUMNS and MAP-PRINT-PAPER
32	72	Device error on terminal; FHS-MAIN-RC is assigned the value 16	Inform system diagnostics staff
32	80	Format not suitable for the terminal; FHS-MAIN-RC is assigned the value 16	Generate format for the correct terminal type
32	84	MAP-PRINT-COLUMNS was changed for a format with ' fast formatting' and PCL printer	Either do not change MAP-PRINT-COLUMNS or generate the format without 'fast formatting'
32	88	The coded character set is not available for 8-bit formats.	Incorporate the coded character set into XHCS.
36	04	FHS-MAPPING-METHOD= “RSET” or “RSON” requested for a partial format that is not on the screen	First output format using FHS-MAPPING-METHOD=“BEGN” or “ONLY”
36	16	Restart area too small	Increase the size of the restart area
36	20	Address of the restart area not found in the MAPLIST area; output formatting without a restart was followed by input formatting with a restart	Check application program; restart should be requested for output formatting as well
36	24	Partial formatting requested and FHS-MAP-PART not filled	Fill FHS-MAP-PART as required
36	28	The MAPLIST area was initialized during an opened partial formatting cycle (output formatting)	Do not enter “Y” in the MAP-CLEAR-OPT field when opening a partial formatting cycle
36	32	MAP-CLEAR-OPT was not supplied with “Y” when a partial formatting cycle (for output formatting) was opened	Enter “Y” in MAP-CLEAR-OPT when opening a partial formatting cycle

ERROR-CATEGORY	ERROR-REASON	Meaning	Action for error recovery
36	40	Specified format was not generated with IFG	Only generate partial formats with IFG
36	44	Specified device type is not supported for partial formatting	Use correct terminal or work without partial formats
36	52	A line to be formatted was already formatted (two formats overlap on the screen)	Check format definition; if necessary redefine start line number
36	56	The start line number has been changed	Check the start line number
36	60	The device type was changed within a partial formatting cycle	Device type must stay the same throughout a partial formatting cycle
36	64	The input node was changed during a partial formatting cycle	Input mode must stay the same throughout a partial formatting cycle
36	68	Different device types for input and output formatting	Do not change device type between input and output formatting
36	72	MAP-SCREEN-PRE-MOD=“Y” can no longer be executed	Do not change MAP-SCREEN-PRE-MOD during the partial formatting cycle. The formatting is being performed as specified for the first partial formatting call in this cycle
36	76	MAP-SCREEN-PRE-MOD was ignored	
36	80	Partial formatting cycle for output not terminated	Terminate partial formatting cycle with FHS-MAP-PART=“L”
36	88	Partial format unknown for input formatting	Repeat output partial formatting cycle
36	92	Start line number was changed during input formatting	Check start line number
36	96	Input formatting has already been performed for this partial format	None; the partial format is formatted again
36	100	No partial format contains any input data from the terminal	None; no formatting has taken place
36	104	A partial format is being formatted without FHS-PARTIAL-MAP-OPT being assigned the value “Y”. A partial format was formatted without the partial formatting function	The partial format is being formatted normally taking account of the start line number

ERROR-CATEGORY	ERROR-REASON	Meaning	Action for error recovery
36	108	No partial formatting in the administrative area (in serial input formatting)	Repeat output formatting cycle
36	112	The value of FHS-RESTART-OPT1 has changed during the current output cycle; for #formats: screen generation was started without the restart function with partial formats having the old structure of the data transfer area	Maintenance of the restart area is mandatory with #formats; avoid mixed operation as far as possible
36	116	Restart execution was requested without prior generation of the restart areas	Only request a restart if a restart area was also used during formatting
36	128	Partial formatting was requested and FHS-MAP-PART="S" specified although the format to be output is not a partial format	Output format without partial formatting or if appropriate assign FHS-MAP-PART="L"
36	132	Partial formatting was requested although the format to be output is not a partial format	None; the format has been formatted normally
36	140	MAPLIST area of old structure and screen dimension not 24x80	Create new MAPLIST area using FHS V6 or later
36	144	Invalid screen dimension	Check format
36	148	Screen dimension was changed in the current output cycle	Use only partial formats having the same screen dimension in the output cycle
48	4	Error in I/O area for input formatting	Check program
48	8	CDS ("code-transmit-key") byte cannot be decoded	Check I/O area
48	12	Format not generated with IFG (format error)	
48	16	Formats with ' fast formatting' cannot be output on a 3287 printer	Generate format without ' fast formatting'
52	4	Incorrect terminal type	
52	8	Terminal type in format is 3287 and value for MAP-PRINTER- CONTROL is incorrect	Use the correct value for MAP-PRINTER-CONTROL
52	12	Terminal type in MAP-DEVICE- CLASS is 3287 and value for MAP-PRINTER-CONTROL is incorrect	Specify the correct value
52	16	Format is longer than one page. A page may have up to 74 lines at 6 lines per inch or 96 lines at 8 lines per inch	Change format specification or reduce the line spacing
52	20	Field length causes a page overflow	Check the format

ERROR-CATEGORY	ERROR-REASON	Meaning	Action for error recovery
56	00	#format could not be formatted	See detailed return code in the control block
56	4	No restart area for #formats for formatting on the data display terminal	Maintain restart area
56	8	Errored field contents in the data transfer area on output formatting (only for fields with edit function)	Check data source, e.g. data base
56	12	Undefined value for a global attribute or invalid combination of global attributes on output formatting	Check global attribute in data transfer area
56	16	Undefined value for a field attribute or invalid combination of field attributes on output formatting	Check field attribute in in data transfer area
56	20	Formatting of a #partial format for data display terminal without partial formatting	Format #partial format with partial formatting
56	28	#format not suitable for variable positioning	Observe restrictions
60	4	Service call cannot be executed; possible causes: service function entry missing in control block or contains an undefined value	Check call
60	8	Service call ' Initialization of Data Transfer Area' cannot be executed because the specified format is not an #format	#formats are prerequisite for this function
64	4	Illegal formatting call, invalid CALL format	Check call
64	8	Error during exit routine loading, invalid exit routine name or module library name	Check name of exit routine or library
64	12	Invalid value specified for A-UPDATE-METHOD	Specify a correct value
68	4	Terminal-specific administrative area was specified during partial formatting or a restart, or is too small	Specify administrative area or increase its size
68	8	Insufficient memory space is available for partial formatting cycle; output formatting is not possible Insufficient memory space is available for serial input formatting of partial formats; input formatting is only possible for the first partial format	Insure there is sufficient memory space

ERROR-CATEGORY	ERROR-REASON	Meaning	Action for error recovery
68	12	Invalid partial format configuration on screen	With DCAM COBOL partial formatting, #partial formats and old partial formats should not appear on the screen simultaneously
80	4	Invalid entry in the data structure FHS-MAIN-PAR	Check FHS-MAIN-PAR
96	0	Unrecognized control character sequence	Check that message has been transferred to data display terminal in its entirety
96	4	Control character sequence for setting the screen dimension cannot be decoded	Check that message has been transferred to data display terminal in its entirety
96	8	The screen dimension specified in the format is not supported by the data display terminal. FHS was unable to detect this as no device status data was transferred	Regenerate format, specifying different screen dimension
96	32	Control character sequence for field display (field attributes) cannot be decoded	Check that message has been transferred to data display terminal in its entirety
96	40/ 52	Characters in a data field can not be displayed using the character set specified in the field attribute; the specified character set was not loaded previously. FHS was unable to detect this as no administrative area was transferred	Check format and regenerate if necessary
96	44	Control character sequence for attribute modification cannot be decoded	Check that message has been transferred to data display terminal in its entirety
96	56/ 88/ 96	Error message from the 9763 Data Display Terminal	Check that message has been transferred to data display terminal in its entirety
96	80	Control character sequence for loading character sets cannot be decoded	Check that message has been transferred to data display terminal in its entirety
96	84	The character set specified in the format cannot be loaded. The required character generator is not present in the device. FHS was unable to detect this as no device status data was transferred	Regenerate format, regenerate character set

ERROR-CATEGORY	ERROR-REASON	Meaning	Action for error recovery
96	92	The character generator required for the character set to be loaded is in use by the data display terminal and not available. FHS was unable to detect this as no device status data was transferred	Regenerate format, regenerate character set file or change assignment of character generators
96	100	The screen dimension specified in the format is not supported by the 9763 Data Display Terminal	Regenerate format, specifying a different screen dimension
96	104	9763 is not specified as the device type in a character set file	Check character set file and regenerate if necessary
96	108	Neither S (single plane) nor T (triple plane) is specified as the character set type in a character set format	Check character set file and regenerate if necessary
96	112	The area for the output message is too small, not all the character sets specified in the format can be loaded	Extend message output area; regenerate format and reduce number of character sets to be loaded
96	116	The characters in a data field cannot be displayed using the character set specified in the field attribute; the specified character set was not loaded previously.	Check format and regenerate if necessary
96	120	A character format, a P key format or a DE format was loaded instead of a format or partial format.	Check format library and format name
96	124	Character set file cannot be decoded	Check character set file and regenerate if necessary
96	128	The character set specified in the format cannot be loaded. The required character generator is not present in the device	Regenerate format, regenerate character set file
96	132	The character generator required for the character set to be loaded is in use by the data display terminal and not available.	Regenerate format, regenerate character set file or change assignment of character generators

Return codes after an FHSINIT call

After an FHSINIT call the following return codes may be issued:

Primary return code

FHS-MAIN-RC	Meaning
0	FHSINIT call was executed without errors
4	Error during FHSINIT processing. The secondary return code in FHS-ERROR-INFO states the nature of the error.
8	Incomplete execution of FHSINIT; FHS has supplied default values in the case of invalid entries.

Secondary return code

ERROR-CATEGORY	ERROR-REASON	Meaning	Action for error recovery
4	4	Loaded format definition(s) is (are) incorrect	Correct or regenerate format definition(s)
8	4	One or more formats cannot be loaded on opening formatting	Check format names and whether the formats are present in the format application file
20	4	Formatting routine MFHSROUT cannot be loaded	Check whether MFHSROUT is present in the specified FHS module library
20	8	Loaded formatting routine MFHSROUT is incorrect	
20	12	Loaded formatting routine MFHSROUT is incompatible with FHS COBOL	Check version number of MFHSROUT
64	4	Illegal FHSINIT call, invalid CALL format	Check call
64	16	An invalid value was specified in FHS-MAP-NO (> 2730 or < FHS-RES-MAP-NO)	Check entries for FHS-MAP-NO and FHS-RES-MAP-NO
80	4	Invalid entry in FHS-INIT-PAR	Check FHS-INIT-PAR
80	8	A value exceeding 100 was specified in FHS-MAP-NO and no memory is available for the directory. FHS-MAP-NO=100 was assumed	Check entries for FHS-MAP-NO and check memory space

After an FHSATTR or FHSCURS call the following return codes may be issued:

Primary return code

FHS-MAIN-RC	Meaning
0	FHSATTR or FHSCURS call was executed without errors
4	Invalid CALL format. FHSATTR or FHSCURS call was not executed
8	FHSATTR or FHSCURS call was executed incompletely

Secondary return code

ERROR-CATEGORY	ERROR-REASON	Meaning	Action for error recovery
64	4	Illegal FHS call, invalid CALL format	Check call
80	4	Invalid entry in FHS-ATTR-PAR (FHSATTR call only)	Check FHS-ATTR-PAR

11.3 Device-specific data

Number of characters per line for each printer type

Printer type	No. chars/line for character spacing		
	1	2	3
9001	80	96	136
9001-31	80	96	120
9001-8931	36	163	204
9003	132	158	198
9004 with continuous paper or single sheets, landscape	36	163	204
9004 with single sheets, portrait	72	86	108
9011-18	80	96	120
9011-19	136	163	204
9012 with single sheets, portrait	80	96	120
9012 with continuous paper	136	163	204
9013 with continuous paper or single sheets, landscape	144	172	216
9013 with single sheets, portrait	76	90	114
9013 with form feed attachment	Paper width infinitely variable		
9022	78	93	117
PCL	78	94	117
3287	132	-	-

Note

The values given are maximum values; they assume an appropriate setting on the device.

The following restrictions exist for the PCL printer 9022-200:
character spacing 1 = 77 characters, character spacing 2 = 93 characters and character spacing 3 = 116 characters. FHS checks only the PCL values; the last column of a line is ignored.

Representation of the display attributes on data display terminals

Display attribute (specified in IFG)	**FHS attribute**	**R epresentation on**			
		8160	**9750, 9751, 9753, 9755, 9763**	**9752**	**3270**
bright	BRT	bright	bright	green	bright
normal	NORM	normal	normal	yellow	normal
invisible	DRK	invisible	invisible	invisible	invisible
flashing	SIGN	flashing	flashing	flashing	--
bright + unterstrichen/kursiv	BRT + ITAL	bright + italics	bright + underlined	red	bright
normal + unterstrichen/kursiv	NORM + ITAL	normal + italics	normal + underlined	white	normal
invers	INVERS	--	inverse[1]	--	--

[1] invers only with 9763

Representation of the FHS attributes on printer terminals

FHS attribute	**Representation on printer**										
	9001	**9002**	**9003**	**9004**	**9011[1]**	**9012**	**9013**	**9022**	**9001-31 9001-8931**	**PCL**	**3287**
NORM	nor-mal	nor-mal	nor-mal	nor-mal	nor-mal	nor-mal	nor-mal	nor-mal	normal	nor-mal	nor-mal
BRT	--[2]	--	--	bold	bold	bold	bold	bold	bold	bold	
SIGN	--	--	--	sha-ded	sha-ded	--	--	sha-ded	shaded	sha-ded	--
ITAL	under lined	italic	italic	under lined	under lined	under lined	under lined	under llined	underlined	italic	under lined
WIDE	wide	wide	wide	simul ated	wide	wide	wide	wide	wide	--	--
TALL	--	tall	--	--	tall	--	--	tall	tall	--	--

[1] With regard to the combination of attributes (e.g. TALL and SIGN), certain important restrictions apply that cannot be taken into consideration by FHS.

[2] The attribute is ignored.

Overview of attributes and associated terminal types and input modes

Attributes	Terminal type and input mode					
	8160/975x		9763		3270	
	I S T D =					
	RMOD	RUNP	RMOD	RUNP	RMOD	RUNP
UNPROT	yes	yes	yes	yes	yes	yes
PROT	yes	yes	yes	yes	yes	yes
PROTRET	yes	yes	yes	yes	yes	yes
FSET	yes	UNPROT	yes	yes	yes	UNPROT
BRT	yes	yes	yes	yes	yes	yes
NORM	yes	yes	yes	yes	yes	yes
DRK	yes	yes	yes	yes	yes	yes
INVERS	ign	ign	yes	yes	ign	ign
PRINT	yes	yes	yes	yes	yes	yes
NOPRINT	yes	yes	yes	yes	yes	yes
SIGN	yes	yes	yes	yes	ign	ign
DET	yes	ign	yes	ign	yes	ign
NUM	yes	yes	yes	yes	yes	yes
IC	yes	yes	yes	yes	yes	yes
ITAL	yes	yes	yes	yes	ign	ign
WIDE	ign	ign	ign	ign	ign	ign
TALL	ign	ign	ign	ign	ign	ign
ASKIP	ign	ign	ign	ign	yes	yes

Key:

UNPROT: The UNPROT attribute is set automatically for the data field.

ign: The attribute is ignored.

yes: The attribute is permitted.

Overview of attributes and associated printer types

Attri-butes	Printer type										
	9003	**9001**	**9002**	**9004**	**9011**	**9012**	**9013**	**9022**	**9001-31 9001-8931**	**PCL**	**3287**
UN-PROT	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign
PROT	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign
PROTRET	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign
FSET	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign
BRT	ign	ign	ign	yes	yes	yes	yes	yes	yes	yes	ign
NORM	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign
DRK	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign
INVERS	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign
PRINT	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign
NO-PRINT	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign
SIGN	ign	ign	ign	yes	yes	ign	ign	yes	yes	yes	ign
DET	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign
NUM	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign
IC	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign
ITAL	yes	und	yes	und	und	und	und	und	und	yes	und
WIDE	yes*	yes	sim	sim	yes	yes	yes	yes	yes	ign	ign
TALL	ign	ign	yes	ign	yes	ign	ign	yes	yes	ign	ign
ASKIP	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign	ign

Key:

ign: The attribute is ignored

yes: The attribute is permitted

sim: The attribute is simulated

und: The attribute means underscore

*: In “fast” formatting, the WIDE attribute is not correctly represented on the 9003 printer.

MDCBL operands for output formatting on different terminals

MDCBL operand	Terminal												
	9002	**9001 9003**	**9001 -31 -8931**	**9004**	**9011**	**9012**	**9013**	**9022**	**8161 975x**	**9763**	**3270**	**PCL**	**3287**
MSTD	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
PMOD	-	-	-	-	-	-	-	-	yes	yes	yes	-	-
BEL	-	yes	yes	yes	-	yes	yes	yes	yes	yes	yes	yes	yes
CLEAR	-	-	-	-	-	-	-	-	yes	yes	yes	-	-
DETC	-	-	-	-	-	-	-	-	-	-	-	-	-
MODY	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
ALLATTR	-	-	-	-	-	-	-	-	yes	yes	yes	-	-
ISTD	-	-	-	-	-	-	-	-	yes	yes	yes	-	-
NILS	-	-	-	-	-	-	-	-	yes	yes	-	-	-
EXIT	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
HCOPY	-	-	-	-	-	-	-	-	yes	yes	-	-	-
AUTOHC	-	-	-	-	-	-	-	-	yes	yes	yes	-	-
KEY-LOCK	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
ATAB	-	-	-	-	-	-	-	-	yes	yes	yes	-	-
DEVICE	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
CNTRLU	yes	yes	yes	yes	yes	yes	yes	yes	-	-	-	yes	yes
EFFLEN	-	-	-	-	-	-	-	-	-	-	-	-	-
UARLEN	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
PAPER	-	yes[1]	yes	yes	yes	yes	yes	yes	-	-	-	yes	yes
HMI	-	yes	yes	yes	yes	yes[2]	yes	yes	-	-	-	yes	yes
VMI	-	yes	yes	yes	yes	yes	yes	yes	-	-	-	yes	yes
PRNTRB	yes	yes	yes	yes	yes	yes	yes	yes	-	-	-	yes	yes
MAPPART	-	-	-	-	-	-	-	-	yes	yes	yes	-	-
RESTART	-	-	-	-	-	-	-	-	yes	yes	yes	-	-
UNLDKE	-	yes[1]	yes	yes	yes	yes	yes	yes	-	-	-	yes	-

[1] For 9003 only, single-sheet feed from one bin

[2] Interpretation of HMI by the printer can be set on the printer

MDCBL operands valid for input formatting on different terminals

MDCBL	Terminal		
	8160, 975x	9763	3270
MSTD		[1]	
PMOD	-	-	-
BEL	-	-	-
CLEAR	-	-	-
DETC	yes	yes	yes
MODY	yes	yes	yes
ALLATTR	-	-	-
ISTD	-	-	yes[2]
NILS	-	-	- [3]
EXIT	yes	yes	yes
HCOPY	-	-	-
AUTOHC	-	-	-
KEYLOCK	-	-	-
ATAB	-	-	-
DEVICE	yes	yes	yes
CNTRLU	-	-	-
EFFLEN	yes	yes	yes
UARLEN	yes	yes	yes
PAPER	-	-	-
HMI	-	-	-
VMI	-	-	-
PRNTRB	-	-	-
MAPPART	-	-	-
RESTART	yes	yes	yes
UNLDKE	-	-	-

[1] MSTD=ONLY is evaluated for all Data Display Terminals, for input formatting as well.

[2] On the 3270 ISTD for input must correspond to the associated output.

[3] On the 3270 an implicit NILS=NO always applies (independently of MDCBL).

Output options for fast formatting

Default value for operation	The format can be output on														
	8161 9750	**9755**	**9763**	**3270**	**9001**	**9001 -31 -8931**	**9002 9003**	**9003**	**9004**	**9011**	**9012**	**9013**	**9022**	**PCL**	**3287**
9750, 8160	yes	yes	yes	no	no	no	no	no	no	no	no	no	no	no	no
9755	no	yes	yes	no	no	no	no	no	no	no	no	no	no	no	no
9763	no	no	yes	no	no	no	no	no	no	no	no	no	no	no	no
3270	no	no	no	yes	no	no	no	no	no	no	no	no	no	no	no
9001	no	no	no	no	yes	yes *)	no	no	no	yes*)	yes*)	no	no	no	no
9001 -31 -8931	no	no	no	no	yes	yes	no	no	no	yes*)	yes*)	no	no	no	no
9002/9003	no	no	no	no	no	yes*)	yes	yes*)	yes*)	yes*)	yes*)	yes*)	yes	no	no
9003	no	no	no	no	no	no	no	yes	no	no	no	no	no	no	no
9004	no	no	no	no	no	no	no	no	yes	no	no	no	yes*)	no	no
9011	no	no	no	no	yes*)	yes*)	no	no	no	yes	no	no	no	no	no
9012	no	no	no	no	no	no	no	no	no	no	yes	no	no	no	no
9013	no	no	no	no	no	no	no	no	no	no	no	yes	no	no	no
9022	no	no	no	no	no	no	no	no	yes	no	no	no	yes	no	no
PCL	no	no	no	no	no	no	no	no	no	no	no	no	no	yes	no
3287	no	no	no	no	no	no	no	no	no	no	no	no	no	no	no

’no’ means: with display terminals: termination of formatting, with a return code
with printers: formatting takes place, the printout may be errored.

*) Restriction: The attributes BRT, SIGN, WIDE, TALL cannot be used; all fields must have the display attribute ’normal’/NORM.

Table of printable characters

Character	Meaning	Character	Meaning
SPACE	(blank, X'40')	%	percent
.	period	_	underline (underscore)
<	less than	>	greater than
(	left parenthesis	?	question mark
+	plus	:	colon
&	ampersand	#	number sign
$	dollar	@	commercial “at”
*	asterisk	'	apostrophe
)	right parenthesis	=	equals sign
;	semicolon	"	double quote
-	minus	a through z	lowercase letters
/	slash	A through Z	uppercase letters
,	comma	0 through 9	numerals

Although the following also count as printable characters, they are not displayed the same on all terminals:

Character	Meaning		
!	exclamation mark		
^	circumflex		
c	cent sign		
$	currency symbol		
[	left bracket	or Ä	(umlauts)
\	backslash	or Ö	
]	right bracket	or Ü	
{	left brace	or ä	
|	vertical bar	or ö	
}	right brace	or ü	

The characters displayed are with reference to EBCDI code. In addition, the following code positions are regarded as valid:

```
X'67' X'8B' X'8C' X'8D' X'AB' X'AC' X'AD'
```

11.4 Generating formats with FHS

11.4.1 Defining formats

The easiest way to define formats (maps) is to use the Interactive Format Generator (IFG). If you wish, however, you can also generate formats using FHS macros.

Exception

#formats, formats for the 9755 and 9763 Data Display Terminals, and 3270 display terminal, and formats for the printers supported as of FHS V6.0 for the first time can only be generated by using IFG.

If you are using IFG, you can skip this section.

In order to define a format (map), the macros MDMAP (Define Map) and MDFLD (Define Field) are required; these are described in the following sections.

The first step is to call the MDMAP macro, at the same time specifying a format name.

The MDFLD macro must be called once for each data field to be defined. It serves to specify the position, attributes and possibly also the name of the field.

FHS fills spaces between the various data fields on the screen with blanks, which cannot be overwritten.

Finally, a further MDMAP macro call is issued in order to terminate the format definition.

The macro call sequence

```
MDMAP
MDFLD
MDFLD
  .
  .
  .
  .
  .
MDFLD
MDMAP
```

must be repeated for each further format to be defined.

11.4.1.1 Initiating and terminating the format definition

MDMAP/KDCMDMAP - define map

Define format

The MDMAP macro has two forms, depending on whether it initiates or terminates a format definition for DCAM/TIAM applications. The KDCMDMAP macro is used for UTM applications.

Form 1

Form 1 of the MDMAP/KDCMDMAP macro initiates a format definition.

Name	Operation	Operands
formatname	MDMAP	TYPE={MAP / DSECT / (MAP,DSECT)} ,DEVICE={8121[/m] / 8122[/m] / 8160}] [,MAPTYPE=SUB[FORM]] [,MODE={IN / OUT / MIX}] [,LAN={ASSEMB / COBOL}] [,CNTRLU={8112 / 8160L}] [,REDEF={variable / NO}] [,REML=x] [,UARFORM={HWAL / UNAL / NOAL}]

Name	Operation	Operands
formatname	KDCMDMAP	TYPE={MAP / DSECT} ,DEVICE={8121[/m] / 8122[/m] / 8160}] [,MODE={IN / OUT / MIX}] [,LAN={COBOL / ASSEMB}] [,CNTRLU={8112 / 8160L}] [,REML=x] [,PREF={char-e/char-a / char}] [,KDCS={1 / 2}]

Meaning of the entries:

formatname Name of the format.
The name may be up to eight characters long.
(Exception: with MODE=MIX,TYPE=DSECT or TYPE=(MAP,DSECT) no more than seven characters).

TYPE=

MAP A format is to be defined.

DSECT An addressing aid is to be generated.

(MAP,DSECT) A format is to be defined and its associated addressing aid generated.

DEVICE= Type of terminal for which the format is defined. The following entries are possible:

```
DEVICE=8121[/m]
DEVICE=8122[/m]
DEVICE=8160
```

m = Number of characters per line

Device	m	Default value
8121 8122 8160	25-132 25-132	132 80 80

MAPTYPE= SUB[FORM]
A subformat is to be defined (optional) (see page 560ff).

Note

The format name in this case may be up to 8 characters; The DEVICE= and MODE= operands can be omitted because FHS will take the relevant information from the main format definition. Addressing aids for COBOL with subformats (SUBFORM=) are not supported.

MODE= IN The addressing aid associated with the format will be generated for input formatting.

OUT The addressing aid associated with the format will be generated for output formatting.

MIX The addressing aid associated with the format will be generated for both input and output formatting (default).

LAN= Specifies the programming language for which an addressing aid will be generated (optional).

ASSEMB ASSEMBLER addressing aid (default for MDMAP)

COBOL COBOL addressing aid (default for KDCMDMAP)

CNTRLU= Specifies the terminal via which the printers are attached as secondary peripherals (locally via a data display terminal, or via a printer controller) (optional). The following entries are possible:

```
CNTRLU=8112
CNTRLU=8160L
```

This operand is mandatory when DEVICE=8121 or 8122.

REDEF=

variable Generates the following statement for COBOL addressing aids (optional):

```
01  name of the addressing aid REDEFINES variable.
```

This permits several COBOL addressing aids to be overlaid. “variable” is a name that conforms to COBOL conventions. The default value is REDEF=NO.

REML= Supports exit routines and specifies the maximum length of the identification fields defined for this format in the MDFLD macro (optional). The following entries are possible:

```
REML=n , n=0,1,...,8
```

When REML=0, no identification fields must be specified (default).

Restriction

REML= must not be specified if the MDMAP macro defines a subformat.

For the use of exit routines see page 286ff.

UARFORM= Structure of COBOL addressing aids for the transfer areas (functions only if TYPE=DSECT and LAN=COBOL) (optional). ASSEMBLER addressing aids are always aligned on halfword boundaries and contain attribute and length fields.

HWAL Transfer areas are aligned on halfword boundaries and contain data and length or attribute fields likewise aligned on halfword boundaries. HWAL is the default value.

UNAL Transfer areas are not aligned, nor do they contain aligned data and length or attribute fields (+formats).

NOAL Transfer areas are not aligned and contain only the data fields (*formats).

Note

UNAL and NOAL can only be used in UTM applications. HWAL is not valid with UTM. The KDCMDMAP macro should be used for UTM applications, the KDCS parameter in this macro being used to define the structure of the transfer areas.

PREF= Specifies which letter the addressing aid is to be prefixed with so that it is unambiguous:

char-e/char-a Specifies the prefix for input and for output to (MODE=MIX).

char Specifies the prefix for input or output to (MODE=IN or MODE=OUT).

This letter is used to distinguish between the addressing aids which would otherwise be identical.

If the operand is not specified, the following default values are assumed:

A for KDCS=2 and MODE=MIX (input)

B for KDCS=2 and MODE=MIX (output)

C for KDCS=1 and MODE=MIX (input)

D for KDCS=1 and MODE=MIX (output)

E for KDCS=2 and MODE=IN or MODE=OUT

F for KDCS=1 and MODE=IN or MODE=OUT

KDCS= Defines how an addressing aid is to be generated:

1 with attribute/length field preceding each field (+format)

2 without attribute/length field (*format)

If no attribute fields are generated, there is no way the field attributes can be modified in the program. When TYPE=MAP this parameter is ignored.

Form 2

Form 2 of the MDMAP/KDCMDMAP macro terminates a format definition.

Name	Operation	Operands
	MCMAP	[TYPE=END]
	KDCMDMAP	

TYPE=END or blank

11.4.1.2 Defining the data fields

MDFLD/KDCMDFLD - define field

Define data fields

The MDFLD macro is used to define a data field in DCAM/TIAM applications. A second form of the macro also permits the insertion of a subformat; see page 560ff. The KDCMDFLD macro is used for UTM applications.

Form 1

Form 1 of the MDFLD/KDCMDFLD macro defines a data field.

Name	Operation	Operands
[name]	MDFLD	[POS={[+]z,[+]l | [+]l}] [,JUST={j | (j,c) | (j,c,j,c)}] [,CONT='text'] [,LEN=m] [,ATTR=attr-list] [,GRPNAM=name] [,EXIT={YES | NO}] [,REM={X'identif' | C'identif'}]
[name]	KDCMDFLD	[POS={[+]z,[+]l | [+]l}] [,JUST={j | (j,c) | (j,c,j,c)}] [,CONT='text'] [,LEN=m] [,ATTR=attr-list] [,GRPNAM=name] [,REM={X'identif' | C'identif'}]

Meaning of the entries:

name Name of the data field (optional)
The name must not exceed 7 characters in length. A data field that is not to be addressed (text field) may not be given a name, as this would reserve space in the addressing aids. Fields whose contents are needed by the application program must be given a name (these are generally fields with the attributes UNPROT, PROTRET, FSET, DET, NUM and IC).

POS= Specifies the position of the data field for output to a terminal.
The entry refers to the first character of the data field (optional).

The following entries are possible:

1. Absolute position

POS=(z,l)
z = line number
l = column number
The first character position on the terminal is POS=(1,1).

2. Relative line and column position

POS=(+z,+l)
z = specifies the number of line feeds to be made.
l = specifies the number of blank columns from the end of the preceding data field.
Caution
z = number of blank lines + 1
l = number of blank columns

3. Relative line position, absolute column position POS=(+z,l)

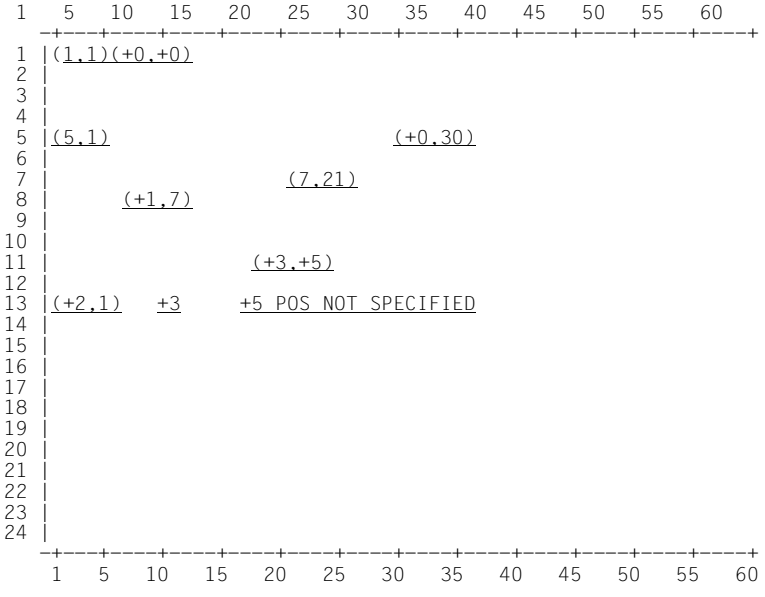
4. A relative position specification within the same line may be written in abbreviated form:

POS=+1 is equivalent to POS=(+0,+1)
Accordingly, POS=+4 means that four character positions are to be left free between the two data fields.

If the operand is omitted, POS=+1 is assumed (one blank between the two data fields). If no position is specified for the first data field, it will start at position (1,2).

Note

If column 80 is occupied in formats, the internal line counter is then also incremented by +1. A blank line is produced. In this case, +0 should be specified for the next line.



Fields are underlined and contain the position made for the field. Relative position specifications refer to the precedent field.

Examples of position specifications

JUST= Controls the alignment of a character string in the data field and defines the fill characters to be used in the field (optional).
This operand applies to both input and output formatting.

For input formatting 'data field' means the field in the transfer area; for output formatting, it refers to the field on the screen.
Thus the data field is in each case the destination field.

Justification (j) and fill character (c) can be specified as follows:

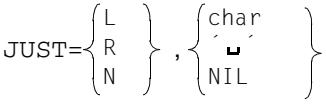
JUST={ L | R | N }

L The data field will be left-justified. The fill character is a blank (X'40'). L is the default value.

R The data field will be right-justified. The fill character is a zero (X'F0').

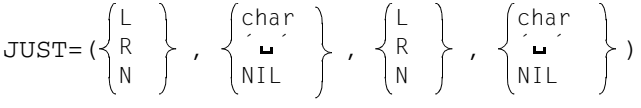
N No data field justification.
The fill character is a blank (X'40').

Note
L can be replaced by B, and R by Z. The effect is the same.



L The data field will be left-justified.
R: The data field will be right-justified.
N: No data field justification.

char: Any single character apart from a blank.
'␣': (Blank enclosed in apostrophes); the fill character is a blank (X'40').
NIL: The fill character is the NULL character (X'00').



These entries have the same meaning as above. The left-hand entries apply to output formatting, the right-hand entries to input formatting (see page 26ff).

CONT= Defines a text field (optional).

'text' 'text' is a fixed text which is to be entered into the data field on every message output. The text must be enclosed in apostrophes and must not exceed 125 characters in length. It may contain any printable character except apostrophe (') or ampersand (&). The application program cannot supply data to fields for which CONT='text' has been specified. The specified text is aligned and filled with fill characters as defined by JUST=. If the CONT operand is omitted, a variable field is defined.

LEN=

m — Specifies the length of a data field (optional).
Any value between 1 and 256 is permitted for “m”. If this operand is omitted, LEN=1 is assumed for variable fields (no CONT operand specified), and the implicit length of the text specified in the CONT operand is used for text fields.

ATTR= See page 268ff.

Note
It is normal practice to use PROT with NORM and UNPROT with BRT.

The following table shows the default values used by FHS:

		with CONT=	without CONT=
without IC and NUM	addressable	P[1]	U[2]
	non-addressable	P	--[3]
with IC or NUM	addressable	U	U
	non-addressable	--	--

[1] P = PROT, NORM, PRINT

[2] U = UNPROT, BRT, PRINT

[3] -- = meaningless

GRPNAM=

name — Combines several fields to form a group (optional).
’name’ signifies a freely selectable name with a maximum of 7 characters. All fields belonging to the group must be defined in immediate succession and must be provided with the GRPNAM=name operand. Two consecutive groups must be separated by at least one field belonging to neither group.

Effect

For group fields, the addressing aid generates a field ’nameA’ or ’nameL’ for the first field of the group only (see page 572).
In this way, several contiguous fields can be treated as a single field in the subsequent processing of the data.
If attributes are changed, the attributes of the first group field apply to the whole group.

EXIT=

NO The field is not to be passed to the exit routine (optional).

YES The field is to be passed to the exit routine.

If the operand is not specified, the following values are assumed:

EXIT=NO
if REML=0 is specified in the MDMAP/KDCMDMAP macro.

EXIT=YES
if a value other than REML≠0 is specified in the MDMAP/KDCMDMAP macro.

If the field defined by MDFLD belongs to a subformat, EXIT=yes must not be specified.

(Exit routine: see page 286ff; MDMAP: see page 545ff).

REM= Identification fields for the exit routine (see page 286ff) (optional).

The following entries are possible:

```
REM=C'.....'
REM=X'.....'
```

The identification field should have the length specified in the REML operand of the MDMAP macro. If not, the field is truncated or padded as follows:

Identification field < REML:
The identification field is padded on the right with X'00' or C'␣'.

Identification field > REML:
The identification field is truncated on the right.

If the operand is omitted, the identification field is set equal to X'00..00'.

REM must not be specified if the field defined by MDFLD belongs to a subformat.

Form 2

Form 2 of the MDFLD macro is used to insert a subformat (see page 560ff).

Name	Operation	Operands
letter	MDFLD	POS=([+]z,[+]l[/{NL | ALL}]),SUBFORM=name[/n]

Meaning of the operands:

letter One alphabetic character
(This letter serves to modify the field names of the addressing aid for the subformat; see page 569ff).

POS=

POS=([+]z,[+]l[/{NL | ALL}])

This operand specifies the point of reference for all position specifications of the subformat. The same entries are permitted as for data fields. If the operand is omitted, POS=(+0,+0) is assumed.

/NL The reference point for the second and following subformats is the beginning of the next line (optional).

/ALL The entry in the POS operand applies to all subformats (only if a relative line specification was used) (optional)

SUBFORM=name

'name' is the name of the desired subformat.

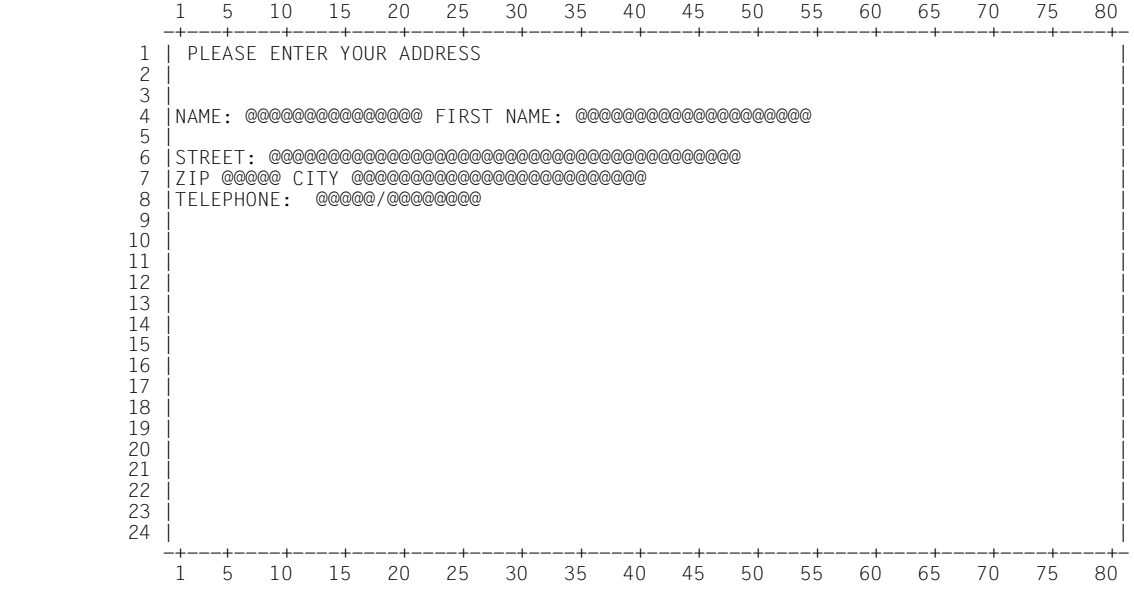
/n Repetition factor (optional). The subformat is to be inserted n times (n=1, ...,10).

Note
If a subformat is to be inserted several times, /NL or /ALL may be specified; otherwise the entry is ignored.

11.4.1.3 Example of a format definition

The terminal operator is to enter his name, address and telephone number on the screen. You define the format ADDRES1 for this purpose:

```
ADDRES1  MDMAP TYPE=MAP,DEVICE=8161
         MDFLD CONT='PLEASE ENTER YOUR ADDRESS'
         MDFLD POS=(4,1),CONT='NAME:'
NAME     MDFLD LEN=15
         MDFLD CONT='FIRST NAME:'
FIRSTNA  MDFLD LEN=20
         MDFLD POS=(6,1),CONT='STREET:'
STREET   MDFLD LEN=40
         MDFLD POS=(7,1),CONT='ZIP'
ZIPCODE  MDFLD LEN=5,ATTR=SIGN
         MDFLD POS=+1,CONT='CITY'
CITY     MDFLD LEN=25
         MDFLD POS=(8,1),CONT='TELEPHONE:'
AREACOD  MDFLD LEN=5,ATTR=NUM
         MDFLD POS=+0,CONT='/'
TNUM     MDFLD POS=+0,LEN=8,ATTR=NUM
         MDMAP
```



Format ADDRES1

11.4.1.4 Generation of formats

Formats can be generated together with the application program or separate from the application program. Generating formats in the application program is suited more to smaller programs having only a few format definitions. The advantages of generation as a separate module are summarized again below:

– the format can be used by different programs,

– the structure and layout of the format can be checked and corrected independently of the associated application program,

– the formats can also be used in application programs with processing sections written in COBOL.

11.4.1.5 Format generation in the application program

Formats are defined by calling the MDMAP and MDFLD macros in the application program. The formats are then assembled together with the application program.

If the operand TYPE=(MAP,DSECT) is specified in the initial MDMAP macro (see page 545ff), a format description and an addressing aid are generated in the form of a DSECT.

Note the following rules:

1. All formats must be defined in the same control section (CSECT).
2. The MGMAP macro must also be called in the same control section and prior to the first format definition.

Name	Operation	Operands
	START	
	MGMAP	.....
	MDMAP	.....
	MDFLD	.....
	.	
	MDFLD	.....
	MDMAP	
	.	
	additional instructions and statements of the application program	
	.	
	.	
	END	

11.4.1.6 Format generation as a separate module

Formats can be defined and assembled separately from the application program.

Proceed as follows:

1. Assemble the macro sequence MDMAP, MDFLD, ..., MDFLD, MDMAP (TYPE=MAP in the first MDMAP call).
2. Transfer the module from the EAM file to a format application file using either the LMR or LMS utility routine.

The START statement can be omitted from the macro sequence; the END statement is mandatory.

The format application file is given the name F.MAPLIB, assuming no other file name has been agreed upon in macro MGMAP (operand MAPLIB=).

Addressing aids associated with the format must be generated in a separate procedure.

The following procedure assembles format definitions (MDMAP and MDFLD macros in the &MAP file) and enters them in the format application file &LIB (default value: &LIB=F.MAPLIB):

```
/BEGIN-PROC LOG=C,PARAM=YES (PROC-PAR=(&MAP,&LIB=F.MAPLIB),ESC-CHAR=C'&')
/ASSIGN-SYSDTA TO-FILE=*SYSCMD
*/START-PROGRAM $ASSEMB
*COMOPT SOURCE=&MAP
*COMOPT MODULE=&LIB
*END HALT
/END-PROC
```

11.4.1.7 Using subformats

Parts of a format can be defined as subformats.

This practice makes sense when certain parts of the format definition are to be inserted a number of times or are to be used in several formats. Thus, for example, a standard header to appear on the screen format can be provided for several formats.

Generation of subformats

Subformats are generated in exactly the same way as formats (main formats), except that the MAPTYPE=SUB[FORM] operand must be specified in the MDMAP macro.

Restrictions

– Subformats can only be generated as a separate module.

– Fields in subformats cannot be processed by EXIT routines.

– In order to insure that the field names of the addressing aid are unique, the first five characters of first-level subformat field names must be unique; for second-level subformats, the first three characters must be unique.

Insertion of subformats

Subformats are inserted into formats by means of form 2 of the MDFLD macro (page 550ff). The name of the subformat must be specified in the SUBFORM= operand of the MDFLD macro.

Any subformat can be inserted into any format.

The format itself can be generated in the application program or as a separate module. When a format is generated in the application program, the names of the associated addressing aids used for the subformats in the program must be specified in the statement:

```
MCALL name1,name2,.....
```

Subformats should always be loaded on opening formatting (MGMAP macro, RESMAP operand) so as to avoid the need to load them subsequently.
It is also possible to insert subformats into another subformat (i.e. second-level subformats). No further levels of nesting are permitted.

Note

Addressing aids for COBOL with subformats (SUBFORM=) are not supported.

Example of the use of a subformat

In the ADDRES1 format, that part which contains the address is to be inserted twice into the ADDRES2 format to allow a second residence to be entered, if applicable.

The format is defined as follows:

```
ADDRES2  MDMAP TYPE=MAP,DEVICE=8161
         MDFLD CONT='PLEASE ENTER YOUR ADDRESS'
         MDFLD POS=(4,1),CONT='NAME:'
NAME     MDFLD LEN=15
         MDFLD CONT='FIRST NAME:'
FIRSTNA  MDFLD LEN=20
A        MDFLD SUBFORM=SUBAD,POS=(6,1)
         MDFLD POS=(10,2),CONT='SECOND RESIDENCE:'
B        MDFLD SUBFORM=SUBAD,POS=(12,1)
         MDMAP
*              SUBFORMAT DEFINITION
*
SUBAD    MDMAP TYPE=MAP,MAPTYPE=SUB
         MDFLD POS=(1,1),CONT='STREET:'
STREET   MDFLD LEN=40
         MDFLD POS=(2,1),CONT='ZIP'
ZIPCODE  MDFLD LEN=5,ATTR=SIGN
         MDFLD POS=+1,CONT='CITY'
CITY     MDFLD LEN=25
         MDFLD POS=(3,1),CONT='TELEPHONE:'
AREACOD  MDFLD LEN=5,ATTR=NUM
         MDFLD POS=+0,CONT='/'
TNUM     MDFLD POS=+0,LEN=8,ATTR=NUM
         MDMAP
```

```
    1   5    10   15   20   25   30   35   40   45   50   55   60   65   70   75   80
   -+---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+-
 1 | PLEASE ENTER YOUR ADDRESS                                                      |
 2 |                                                                                |
 3 |                                                                                |
 4 |NAME: @@@@@@@@@@@@@@@ FIRST NAME: @@@@@@@@@@@@@@@@@@@@                           |
 5 |                                                                                |
 6 |STREET: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@                                |
 7 |ZIP @@@@@ CITY @@@@@@@@@@@@@@@@@@@@@@@@@                                        |
 8 |TELEPHONE:  @@@@@/@@@@@@@@                                                      |
 9 |                                                                                |
10 |SECOND RESIDENCE:                                                               |
11 |                                                                                |
12 |STREET: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@                                |
13 |ZIP @@@@@ CITY @@@@@@@@@@@@@@@@@@@@@@@@@                                        |
14 |TELEPHONE:  @@@@@/@@@@@@@@                                                      |
15 |                                                                                |
16 |                                                                                |
17 |                                                                                |
18 |                                                                                |
19 |                                                                                |
20 |                                                                                |
21 |                                                                                |
22 |                                                                                |
23 |                                                                                |
24 |                                                                                |
   -+---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+-
    1   5    10   15   20   25   30   35   40   45   50   55   60   65   70   75   80
```

Format ADDRES2

11.4.2 Addressing aids for the transfer areas in the application program Addressing the data fields

The format definition macros provide you with symbolic addresses for the fields of the transfer area, where the application program fetches and supplies the data.

An addressing aid assigns names to the fields of the transfer area; in other words, it describes (defines) the latter's structure.
The format definition includes a name (fieldname) for each field that you wish to address.

Fields for input formatting are addressed with

```
fieldnameI
```

Fields for output formatting are addressed with

```
fieldname0
```

Example

After input formatting, the contents of the field NAME are addressed by means of NAMEI.

Each data field is preceded by a 2-byte field (as defined) which is also addressable:

```
fieldnameL   Length field   (input formatting)
```

Prior to input formatting, this field must be supplied with the attribute value of the previous output formatting process. If attributes have been changed, FHS takes the information from this field.

Following input formatting, this field contains the number of characters entered in the data field.

```
fieldnameA   Attribute field   (output formatting)
```

When attributes are to be changed, FHS takes the information from this field (for output formatting, see page 268ff).

You define your own transfer area in your (ASSEMBLER) application program by means of the macro calls

`formatnameI` and `formatname0`

These calls generate a sequence of DS statements which are used to define the input transfer area (formatnameI) and output transfer area (formatname0).
For COBOL addressing aids see page 404ff.

If you are generating your formats with IFG, you can skip the rest of this chapter.

11.4.2.1 Generating addressing aids - defining the transfer areas

The addressing aids can be generated together with the application program or separate from the application program.

– when generated together with the application program, you receive the addressing aids as DSECTs which must be superimposed as overlays on the transfer areas. You must define the transfer areas yourself (see page 565).

– when generated separate from the application program, you are provided with macros which generate DS statements and thus the transfer areas themselves (see page 566).

The same macros that were used to define the format (see page 544ff) are also employed to create the associated addressing aids. The user specifies in the MODE operand of the MDMAP macro whether the addressing aid is to be created for input or output formatting or for both. If MODE=MIX (for input and output formatting) is specified, two separate addressing aids are generated so that input and output data can be better distinguished. If a format contains input data only or output data only, MODE=IN or MODE=OUT can be used as required (cf. also IN and OUT operands of the MCMAP macro).

Note that with MODE=MIX the name may be no more than 7 characters long.

11.4.2.2 Generation of addressing aids in the application program

You generate the format and specify TYPE=(MAP,DSECT) in the MDMAP macro. (The macro sequence then creates the format definition and the associated addressing aids.) The latter are created as DSECTs.

FHS expects the addresses of the DSECTs to be contained in the following symbolic registers:

```
MMAPREGI      for input formatting and
MMAPREGO      for output formatting.
```

The register numbers ‘ri’ and ‘ro’ must not be identical.

In your application program you must

– assign the actual register numbers by means of EQU statements

```
MMAPREGI EQU  ri           Register for input DSECT
MMAPREGO EQU  ro           Register for output DSECT
```

– load the symbolic registers with the addresses of the input and output transfer areas.

– define the input and output transfer areas by means of DS statements. You must specify the length of these areas yourself in accordance with the length and number of your data fields. The areas must be aligned on a halfword boundary.

The DSECTs are laid over the transfer areas and you can address the fields using the names in the addressing aid.

Example

```
FHSPROG  START
MMAPREGI EQU   5
MMAPREGO EQU   6
         MGMAP IOAREA=INOUT,IOLEN=1000,CSTM=RTIO
*
EIN      DS    0H
         DS    CL131
AUS      DS    0H
         DS    CL131
**************************************
*        FORMAT DEFINITION WITH      *
*        TYPE=(MAP,DSECT)            *
**************************************
ANF      BALR  3,0
         USING *,3
         LA    MMAPREGI,EIN
         LA    MMAPREGI,AUS
           -
           -
```

11.4.2.3 Generation of addressing aids separate from the application program

The same macros are used as for the format definition, except that TYPE=DSECT must be specified in the MDMAP call.

You now have to

– assemble the MDMAP/MDFLD macro sequence

– call the utility routines MMAINT (see page 576ff) and MLU to transfer the addressing aid from the module in the EAM file to a macro library.

MMAINT reads the assembled addressing aids from the EAM file and writes them to a work file, F.MAC.INPUT. This file must be allocated as input file for the MLU utility routine. MLU places the addressing aids in the private macro library F.MACLIB.

Example

(as a procedure; the macro sequence is contained in the file &DSECT)

```
/BEGIN-PROC LOG=C,PARAM=YES (PROC-PAR=(&DSECT))
/DEL-SYS-FILE FILE-NAME=OMF
/ASSIGN-SYSDTA TO-FILE=&DSECT
/START-PROGRAM $ASSEMB
/START-PROGRAM $MMAINT
/ASSIGN-SYSDTA TO-FILE=F.MAC.INPUT
/START-PROGRAM $MLU
/END-PROC
```

Note

MMAINT also produces the control statements for the MLU utility routine. To convey the addressing aids into a private macro library with a different name, the first record in the F.MAC.INPUT file must be changed before the MLU utility routine can be called.

The addressing aid for the format is then available as a macro in the private macro library F.MACLIB.

11.4.2.4 Calling the generated addressing aids

The call

```
formatnameI
```

generates a sequence of DS instructions and thus defines the input transfer area.

Similarly, the call
```
formatnameO
```
defines the output transfer area.

If MODE=IN or MODE=OUT was specified in the MDMAP macro, the call
```
formatname
```
is sufficient,

where
```
formatname
```
is the name of the format specified in the MDMAP macro or the name of the addressing aid, if it is different.

Note

Before you assemble your program, you need to assign the private macro library, which must also contain the FHS macros, with the commands

```
/SET-FILE-LINK FILE-NAME=filename, LINK-NAME=ALTLIB /PARAM ALTLIB=yes
```

(filename: name of the private macro library).

Example

The call **ADDRES1I** generates the following statements:

```
ADDRES1I DS    0H
NAMEL    DS    H
NAMEI    DS    CL15
FIRSTNAL DS    H
FIRSTNAI DS    CL20
STREETL  DS    H
STREETI  DS    CL40
ZIPCODEL DS    H
ZIPCODEI DS    CL5
CITYL    DS    H
CITYI    DS    CL25
AREACODL DS    H
AREACODI DS    CL5
TNUML    DS    H
TNUMI    DS    CL8
```

Example

The call **ADDRES1O** generates the following statements:

```
ADDRES1O DS    0H
NAMEA    DS    H
NAMEO    DS    CL15
FIRSTNAA DS    H
FIRSTNAO DS    CL20
STREETA  DS    H
STREETO  DS    CL40
ZIPCODEA DS    H
ZIPCODEO DS    CL5
CITYA    DS    H
CITYO    DS    CL25
AREACODA DS    H
AREACODO DS    CL5
TNUMA    DS    H
TNUMO    DS    CL8
```

11.4.2.5 Addressing aids for subformats

If a format incorporates a subformat, the associated addressing aid must be generated as a separate module. The steps to be taken are the same as described on page 566.

As subformats can be inserted at different points, and a number of times, their field names are modified.

B	**0**	**STREE**	**I**
Letter in name field of MDFLD macro which inserts the subformat.	Repetition factor (0 to 9, if the subformat with /n is to be inserted several times at the same point).	Field name, truncated to 5 (or 3) characters	Data field for input formatting

Example

Addressing aid for the ADDRES2 format (input formatting)

```
ADDRES2I DS    0H
NAMEL    DS    H
NAMEI    DS    CL15
FIRSTNAL DS    H
FIRSTNAI DS    CL20
AOI      SUBAD L,I
AOSTREEL DS    H
AOSTREEI DS    CL40    STREET
AOZIPCOL DS    H
AOZIPCOI DS    CL5     ZIP CODE
AOCITYL  DS    H
AOCITYI  DS    CL25    CITY
AOAREACL DS    H
AOAREACI DS    CL5     AREA CODE
AOTNUML  DS    H
AOTNUMI  DS    CL8     TELEPHONE NUMBER
BOI      SUBAD L,I
BOSTREEL DS    H
BOSTREEI DS    CL40    STREET
BOZIPCOL DS    H
BOZIPCOI DS    CL5     ZIP CODE
BOCITYL  DS    H
BOCITYI  DS    CL25    CITY
BOAREACL DS    H
BOAREACI DS    CL5     AREA CODE
BOTNUML  DS    H
BOTNUMI  DS    CL8     TELEPHONE NUMBER
```

If a subformat is inserted into another subformat, the names of the second-level subformat are twice modified.

Note

Addressing aids for COBOL with subformats (SUBFORM=) are not supported.

Example

The address is to be inserted into the format ADDRES3 as a subformat and the telephone number twice into the address as a subformat.

```
ADDRES3  MDMAP TYPE=MAP,DEVICE=8161
         MDFLD CONT='PLEASE ENTER YOUR ADDRESS'
         MDFLD POS=(4,1),CONT='NAME:'
NAME     MDFLD LEN=15
         MDFLD CONT='FIRST NAME:'
FIRSTNA  MDFLD LEN=20
A        MDFLD SUBFORM=SUBA,POS=(6,1)
         MDMAP
SUBA     MDMAP TYPE=MAP,MAPTYPE=SUB
         MDFLD POS=(1,1),CONT='STREET:'
STREET   MDFLD LEN=40
         MDFLD POS=(2,1),CONT='ZIP'
ZIPCODE  MDFLD LEN=5,ATTR=SIGN
         MDFLD POS=+1,CONT='CITY'
CITY     MDFLD LEN=25
U        MDFLD POS=(3,1),SUBFORM=TELE
         MDFLD POS=+3,CONT='WORK'
V        MDFLD POS=(4,1),SUBFORM=TELE
         MDFLD POS=+3,CONT='HOME'
         MDMAP
TELE     MDMAP TYPE=MAP,MAPTYPE=SUB
         MDFLD POS=(1,1),CONT='TELEPHONE'
AREACOD  MDFLD LEN=5,ATTR=NUM
         MDFLD POS=+0,CONT='/'
TNUM     MDFLD POS=+0,LEN=8,ATTR=NUM
         MDMAP
```

Format ADDRES3:

```
    1   5   10   15   20   25   30   35   40   45   50   55   60   65   70   75   80
   -+---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+-
 1 | PLEASE ENTER YOUR ADDRESS                                                      |
 2 |                                                                                |
 3 |                                                                                |
 4 |NAME: @@@@@@@@@@@@@@@@ FIRST NAME: @@@@@@@@@@@@@@@@@@@@@@                      |
 5 |                                                                                |
 6 |STREET @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
 7 |ZIP @@@@@ CITY @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@|
 8 |TELEPHONE:  @@@@@/@@@@@@@@    WORK                                             |
 9 |TELEPHONE:  @@@@@/@@@@@@@@    HOME                                             |
10 |                                                                                |
11 |                                                                                |
```

The addressing aid for ADDRES3 (input formatting) is then as follows:

```
ADDRES3I DS    0H
NAMEL    DS    H
NAMEI    DS    CL15
FIRSTNAL DS    H
FIRSTNAI DS    CL20
AOI      SUBA  L,I
AOSTREEL DS    H
AOSTREEI DS    CL40    STREET
AOZIPL   DS    H
AOZIPI   DS    CL5     ZIP CODE
AOCITYL  DS    H
AOCITYI  DS    CL25     CITY
UOI      TELE  L,I
UOAOARCL DS    H
UOAOARCI DS    CL5     AREA CODE
UOAOTNULL DS    H
UOAOTNUI DS    CL8     TELEPHONE NUMBER
VOI      TELE  L,I
VOAOARCL DS    H
VOAOARCI DS    CL5     AREA CODE
VOAOTNULL DS    H
VOAOTNUI DS    CL8     TELEPHONE NUMBER
```

Note

The first five characters of field names in subformats (first three in the case of second-level subformats) must be unique since the names are truncated during modification (e.g. UOAOTNUI in the preceding addressing aid).

11.4.2.6 Addressing aids for group fields

When several fields are combined into a group (GRPNAM= name operand in the MDFLD macro, page 550), only one length field and attribute field are defined for all fields in the group.

– The length field contains the length of the first field of the group.

– The attribute field applies to all fields of the group.

Example

Addressing aid for the group fields field1, field2,..., fieldn with the group name grpname (output formatting).

```
                    .
                    .
grpnameA            DS          H
grpnameO            DS          OC
field1O             DS          CLn1
field2O             DS          CLn2
                    .
                    .
fieldnO             DS          CLnn
                    .
                    .
```

11.4.2.7 Addressing aids in COBOL

If the processing section of the user's application program is written in COBOL (e.g. in order to make use of it in conjunction with FHS COBOL calls), the addressing aids must also be written in COBOL. IFG includes a function for this purpose. If formats are generated by means of the FHS macros MDMAP and MDFLD, the following applies:

Data items are addressed in COBOL by the same names as in ASSEMBLER.

COBOL addressing aids must be generated as a separate module, as described on page 564ff, except that LAN=COBOL must be specified in the MDMAP macro. Addressing aids are transferred to the source program library by means of MMAINT, MMAINTCB and COBLUR.

The sample procedure below illustrates how COBOL addressing aids are compiled and transferred to the source program library F.COBLIB (the macro sequence is stored in the &DSECT file).

```
/BEGIN-PROC LOG=C,PARAM=YES (PROC-PAR=(&DSECT))
/DEL-SYS-FILE FILE-NAME=OMF
/ASSIGN-SYSDTA TO-FILE=&DSECT
/START-PROGRAM $ASSEMB
/START-PROGRAM $MMAINT
/ASSIGN-SYSDTA TO-FILE=F.MAC.INPUT
/START-PROGRAM $MMAINTCB
/ASSIGN-SYSDTA TO-FILE=F.COB.INPUT
/START-PROGRAM $COBLUR
/END-PROC
```

If the FHS macros MDMAP and MDFLD are not in the macro library $TSOS.MACROLIB, you have to assign your private macro library in the procedure with the following commands:

```
/SET-FILE-LINK FILE-NAME-=filename,LINK-NAME=ALTLIB /PARAM ALTLIB=yes
```

Note

MMAINTCB also generates the control statements for COBLUR. If the source program library is to have a name other than F.COBLIB, the first record of the F.COB.INPUT file must be modified prior to calling COBLUR.

Addressing aids are copied into the COBOL program by means of

`COPY 'formatnameI'.` or `COPY 'formatname0'.`

Addressing aids for COBOL with subformats (SUBFORM=) are not supported.

Formats generated using IFG are copied by means of

```
01  formatname.
    41 length-field PIC 9(n) COMP.
    COPY  [x]formatname.
```

where x (optional) is the prefix specified in IFG,

n = 4 for DCAM
n = 5 for TIAM

CAUTION!

- If COBOL programs are called as subprograms of ASSEMBLER programs, the COBOL addressing aids must generally be defined in the LINKAGE-SECTION. If the addressing aids are in the LINKAGE-SECTION, a transfer area must also be defined (i.e. an addressing aid generated) in the ASSEMBLER program.
- If the addressing aid is actually located in the COBOL section, you need to use a DSECT in the ASSEMBLER section.

Example

COBOL addressing aid for the format ADDRES1

```
for input:

01  ADDRES1I.
    02  NAMEL PICTURE S9(4) COMP SYNC.
    02  NAMEI PICTURE X(15).
    02  FIRSTNAL PICTURE S9(4) COMP SYNC.
    02  FIRSTNAI PICTURE X(20).
    02  STREETL PICTURE S9(4) COMP SYNC.
    02  STREETI PICTURE X(40).
    02  ZIPL PICTURE S9(4) COMP SYNC.
    02  ZIPI PICTURE X(5).
    02  CITYL PICTURE S9(4) COMP SYNC.
    02  CITYI PICTURE X(25).
    02  ARCL PICTURE S9(4) COMP SYNC.
    02  ARCI PICTURE X(5).
    02  TNUML PICTURE S9(4) COMP SYNC.
    02  TNUMI PICTURE X(8).
```

```
for output:

01  ADDRES1O.
    02  NAMEA PICTURE S9(4) COMP SYNC.
    02  NAMEO PICTURE X(15).
    02  FIRSTNAA PICTURE S9(4) COMP SYNC.
    02  FIRSTNAO PICTURE X(2O).
    02  STREETA PICTURE S9(4) COMP SYNC.
    02  STREETO PICTURE X(4O).
    02  ZIPA PICTURE S9(4) COMP SYNC.
    02  ZIPO PICTURE X(5).
    02  CITYA PICTURE S9(4) COMP SYNC.
    02  CITYO PICTURE X(25).
    02  ARCA PICTURE S9(4) COMP SYNC.
    02  ARCO PICTURE X(5).
    02  TNUMA PICTURE S9(4) COMP SYNC.
    02  TNUMO PICTURE X(8).
```

11.4.2.8 Utility routines for generation of addressing aids for formats created using macros

During assembly the addressing aids are first stored in the corresponding result file (EAM file) like any assembled program. The regular utility routines do not offer the functions required for transferring addressing aids from the EAM file to a macro or source program library. The necessary functions are provided by the MMAINT and MMAINTCB utilities.

Utility Routine MMAINT

The MMAINT utility is required when generating addressing aids off line for ASSEMBLER and COBOL programs (see page 563ff).

MMAINT edits the assembler output in the EAM file so that it can then be processed further by MLU for ASSEMBLER addressing aids, or MMAINTCB for COBOL addressing aids. MMAINT stores its results in a file, F.MAC.INPUT. The contents of this file subsequently serve as input for MLU or MMAINTCB.

Note

If MMAINT is started from a data display terminal, the first thing to appear on the screen is the message:

```
'MMAINT PROGRAM VERSION XXX'
```

where XXX is the three-digit version number of the MMAINT utility.

Error messages are output on SYSOUT.

They have the following format: **34xx_[additional-information]**

Identifier	Additional information	Meaning
3412	None	EAM file empty
3413	First 18 characters of the control statement First 18 characters of the module statement	Module name more than 8 characters long Macro name more than 8 characters long Source program name more than 8 characters long
3422	None	Hardware error in disk storage unit
3426	First 18 characters of the module statement	Incorrect library format
3427	None	The module contained no macro definition or source program
3428	Name of macro or source program	Error during compression of macro or source program
3430	Name of macro or source program	Error during assembly: The macro was not concluded with a MEND statement The source program was not concluded with a SEND statement

Utility Routine MMAINTCB

This utility is required only to create COBOL addressing aids.

MMAINTCB processes the information placed in the file F.MAC.INPUT by MMAINT so that it can be used as input for the COBLUR utility routine. MMAINTCB places its output for COBLUR in the file F.COB.INPUT.

Note

When MMAINTCB is started from a data display terminal the first message to appear is:

```
'MMAINTCB PROGRAMM VERSION XXX'
```

where XXX is the three-digit version number of the MMAINTCB utility.

11.5 Tables

11.5.1 Correlation of attributes

FHS attribute field	**Field attribute for #formats**	**IFG**	**COBOL CALL “FHSATTR”**
UNPROT PROT DET + PROT	PROTECTION= “UNPROTECTED” “PROTECTED” “DETECTABLE”	unprotected protected protected, detectable	A-PROT-LEVEL= “UNPR” “PROT” “PDET”
PROTRET	PROTECTION= “PROTECTED” INPUT CONTROL= “AUTORET IN”	protected, automatic input	A-PROT-LEVEL= “PRET”
FSET	PROTECTION= “UNPROTECTED” INPUT CONTROL= “AUTORET IN”	unprotected, detectable	A-PROT-LEVEL= “FSET”
BRT NORM	INTENSITY= “HIGH INTENSITY” “NORMAL INTENSITY”	bright normal	A-DISP-LEVEL=“B” A-DISP-LEVEL=“N”
--- SIGN DRK	VISIBILITY= “VISIBLE” “SIGNALING” “INVISIBLE”	--- flashing invisible	--- A-SIGNAL=“Y” A-DISP-LEVEL=“D”
ITAL	UNDERLINE= “UNDERLINED”	underscored/italic	A-ITAL=“Y”
INVERS	INVERSE= “INVERSE”	inverse	---
IC	CURSOR= “CURSOR”	cursor	---
NUM TALL WIDE	--- --- ---	NUM lock --- ---	A-NUMERIC=“Y” A-TALL=“Y” A-WIDE=“Y”
ASKIP	PROTECTION= “ASKIP”	protected, numeric, skipped by the cursor, non-detectable	A-ASKIP= “Y”

11.5.2 Data formats output/expected by FHS according to the access method

TIAM (RTIO)

During output formatting, FHS generates messages that can be output by means of WROUT or WRTRD; for input formatting, FHS expects messages as supplied by WRTRD. It is important to specify the operand MODE=FORM (see the "TIAM (TRANSDATA, BS2000)" User Guide for details).

Examples

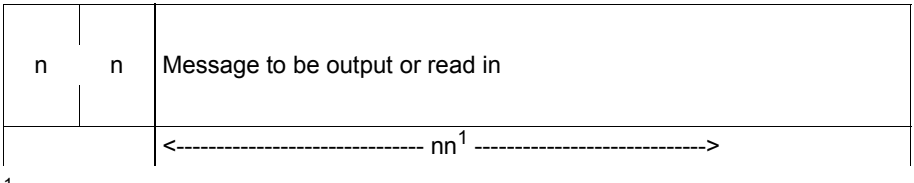
```
MGMP    IOAREA=INOUT,CSTM=RTIO,...

MCMAP   ....,OUT,...
WROUT   INOUT,...,MODE=FORM
```

or

```
MCMAP   ....,OUT,...
WRTRD   INOUT,,INOUT,....,MODE=FORM
MCMAP   ....,IN
```

DCAM

During output formatting, FHS generates messages in the following format; for input formatting, FHS expects messages having the following format in the physical input/output area.



The length of the message should be transferred from the physical input/output area to the RPB control block before output with YSEND, or or transferred ahead of the message read in in the physical input/output area after input with YRECEIVE.

The message without the length field should always be specified as the message area for DCAM (i.e. IOAREA + 2) (see the "DCAM (TRANSDATA)" User Guide for details). The CCB operands EDIT=SYSTEM, EDITIN=(FORM,LCASE) and EDITOUT=FORM are important, HCOPY is not permitted.

Glossary

ADDPOP

Generation of dialog boxes.

addressing aid

An addressing aid describes the data structure in the data transfer area. It allows the programmer to reference the fields of a format by means of symbolic names when the format is used.

attribute

Characteristic relating to the display, editing or checking of a format or field. An attribute is defined either during format generation using IFG (static attribute) or in the application program by way of the global and field attributes (dynamic attributes).

attribute field

Field in the addressing aid in which the attributes of the fields can be entered.

basic format

Format output by the application and not overlaid by dialog boxes.

box

Abbreviation of dialog box; see relevant entry.

bypass mode

Operating mode for a printer connected locally to a data display terminal, where a message is printed without being displayed on the data display terminal.

character set file

File containing the character sets generated using ICE.

control block

Memory area used to store formatting parameters and acknowledgments.

data transfer area (message area)
This area contains the fields and attributes that are accessible to the program. The area is used for the exchange of data between the application program and FHS when the format is used.

DE format
Format that can use the functions of the dialog extension. The attribute "DE format", must be explicitly specified during generation with IFG.

dialog box
Square frame on the screen which contains a DE format.

dialog extension
Component of FHS which allows formats conforming to the Alpha Style Guide to be displayed on the screen. Dialog extension enables multilevel intermediate dialog, command input, extended input checks, and an application-specific help system and messages, amongst other things.

differential output
Output of a format in which only those fields that have been changed by the application program are output afresh on the data display terminal.

exclusion character
Character on the screen which indicates a locked selection of a selection field.

exit routine
User-written routine that checks the fields of a format for particular contents.

explicit box
Dialog box that is output by the application.

fast formatting
see *preformatting*

format
Logical data structure that describes a "form".

format application file
Library used to store the format definitions.

global help
Help for the objects of a DE format that consist of several components, such as single-choice or multiple-choice selection fields.

hardcopy mode
Operating mode for a printer connected locally to a data display terminal, where a message that is displayed on the data display terminal is also output on the printer.

help box
Dialog box containing help information; output by FHS.

implicit box
Dialog box output by FHS, e.g. for messages or help information.

input field
Field in which data for the application program is entered by the terminal operator.

input formatting
During input formatting FHS selects the fields that have been modified and detected by the terminal user from the terminal-specific input message and supplies the application program with the field contents in the data transfer area.

KEY format
Format containing the assignment of function keys.

MAPLIST area
Administrative area required by FHS when using partial formats.

modal box
Dialog box that expects an entry from the user. The underlying box or format is inactive.

modeless box
Dialog box that does not expect direct user action. The underlying box or format remains active.

multiple field mode
Function that permits more fields per line in formats when the data display terminal is set to ‘Operation with a field control character’.

output field
Field into which data is output by the application program.

output formatting

During output formatting FHS inserts the fields from the application program (data transfer area) into the format, i.e. FHS generates a terminal-specific output message that, when output, represents the "completed form" on the terminal.

partial format

Format occupying only a defined part of the screen.

physical input/output area

Area containing the device-specific message for a data display terminal or a printer.

preformatting

(or also “fast formatting”)
The device protocol is contained as far as possible in the format definition. As a result, only parts of the message have to be generated dynamically in order to print the format with the path lengths to be followed in FHS being shortened. The disadvantage of this method is that these formats can only be output on one device type and possibly on upwards compatible devices.
Whether or not a format is preformatted is determined when the format is created in the IFG.

presentation image (PI)

Save area that contains all the information that must be transferred from the output to the input formatting; saved by UTM.

printer acknowledgment

Acknowledgment (positive or negative) that is output on the data display terminal. If a printer is connected in bypass mode, the acknowledgment is issued to the application program.

program unit

Subprogram generated for a UTM application.

REMPOP

Removes boxes; the previous background is displayed again.

restart area

Area required for restarting a screen.

screen restart

Fresh output of the most recent, completely formatted screen after an interruption.

text field

Field containing fixed text that is defined during format generation.

VTSU code

Standard control character that can be added to a file or a message in order to address specific printer functions. RSO recognizes VTSU codes and translates them into printer control characters for the relevant destination printer.

Related publications

Ordering manuals

The manuals are available as online manuals, see *http://manuals.fujitsu-siemens.com*, or in printed form which must be paid and ordered separately at *http://FSC-manualshop.com*.

[1] **FHS V8.1A (BS2000/OSD, TRANSDATA)**
Dialog Extension for TIAM and SDF-P
User Guide

Target group
Application developers
Contents
The manual describes the program interface for using the FHS dialog manager in TIAM and SDF-P applications.

[2] **IFG V8.3A** (BS2000/OSD)
IFG for FHS
User Guide

Target group
Terminal users, application engineers and programmers
Contents
The Interactive Format Generator (IFG) is a system that permits simple, user-friendly generation and management of formats at a terminal. In conjunction with FHS, these formats can be used on the host computer. This user guide describes how formats are generated, modified and managed.

[3] **openUTM V5.2** (BS2000/OSD, UNIX, Windows)
Administering Applications
User Guide

Target group
This manual is intended for everyone responsible for administering openUTM applications and generating administration programs.
Contents
The manual describes the program interface to administration, which enables you to generate your own administration programs. It also describes the command interface to administration and the options available for the administration of message queues and printers.

[4] **openUTM V5.2** (BS2000/OSD, UNIX, Windows)
Generating Applications
User Guide

Target group
This manual is designed for use by application planners and developers as well as operators of UTM applications.
Contents
This manual describes how to define the configuration for a UTM application using the UTM tool KDCDEF and how to create the KDCFILE. One chapter also goes into more detail about the generation of selected objects and functions of the application.
Additional topics include the dynamic configuration of an application and the updating of the KDCFILE using the tool KDCUPD.

[5] **openUTM V5.2** (BS2000/OSD, UNIX, Windows)
Programming Applications with KDCS for COBOL, C and C++
User Guide

Target group
This manual is intended for programmers who wish to use the KDCS program interface for programming UTM applications.
Contents
The manual describes the KDCS interface in the form valid for COBOL, C and C++. This interface incorporates both the basic functions of the Universal Transaction Monitor and the calls for distributed processing. It also contains a description of working together with databases.

[6] **openUTM V5.2** (BS2000/OSD)
Messages, Debugging and Diagnostics
User Guide

Target group
This manual is intended for programmers, generators and administrators of UTM applications in BS2000/OSD.
Contents
The manual describes the debugging of UTM applications, the format of UTM dumps, behavior in the event of errors, and the openUTM message concept. It also includes all messages and return codes output by openUTM.

[7] **RPG3** (BS2000)
RPG Compiler
User's Guide

Target group
RPG users in BS2000
Contents
- Calling and controlling the RPG3 compiler
- Input and compilation of source programs
- Generation and management of object and load modules
- Controlling program execution
- Runtime error handling
- File processing
- Terminal mapping support (FHS/IFG interface)
- Language interfacing (COBOL, assembler)
- /COPY statement
- DMS-Monitor

[8] **TIAM** (TRANSDATA, BS2000)
User Guide

Target group
- BS2000 users (non-privileged)
- Programmers

Contents
- All TIAM commands and macros
- The TIAM COBOL interface with the TIAM COBOL macros
- Examples

Applications
BS2000 timesharing mode

[9] **DCAM** (TRANSDATA)
Program Interfaces
Reference Manual

Target group
– Managers
– Application planners
– Programmers
– System and network administrators

Contents
Description of the Data Communication Access Method DCAM

[10] **DCAM** (TRANSDATA)
COBOL Calls
User Guide

Target group
Programmers of DCAM COBOL programs

Contents
– Special techniques for the use of DCAM COBOL calls, data structures and communication areas
– DCAM COBOL calls, arranged according to their functions
– Examples, programs and program drafts

[11] **BS2000/OSD-BC**
Executive Macros
User Guide

Target group
The manual addresses all BS2000/OSD assembly language programmers.

Contents
The manual contains a summary of all Executive macros, detailed descriptions of each macro with notes and examples, including job variable macros, and a comprehensive general training section.

[12] **BS2000**
User Commands (ISP Format)
User Guide

Target group
BS2000 users (nonprivileged)
Contents
– All BS2000 system commands in alphabetical order with detailed explanations and examples
– The following products are dealt with: BS2000-GA, MSCF, JV, FT, TIAM
Applications
BS2000 interactive/batch mode, procedures

[13] **RSO V3.4A** (BS2000/OSD)
Remote SPOOL Output
User Guide

Target group
This manual is directed at nonprivileged users, RSO device administrators, SPOOL administrators and systems support of BS2000/OSD.
Contents
The manual describes the functions and options of the user groups with respect to utilizing and controlling decentralized printers (RSO printers) and deals with the technical characteristics of all RSO printers.

[14] **Style Guide**
Guidelines on the Design of User Interfaces
User's Guide

Target group
Developers of application programs
Contents
The Style Guide contains rules and recommendations for the development of uniform user interfaces. It describes their structure and contents, and how they are used.

[15] **XHCS**
(BS2000/OSD)
8-bit Code and Unicode Support in BS2000/OSD
User Guide

Target group
Application programmers and system administrators
Contents
XHCS (Extended Host Code Support) is a software package of BS2000/OSD that lets you use extended character sets and/or the Unicode character set in conjunction with 8-bit terminals. XHCS is also the central source of information on the coded character sets in BS2000/OSD.

[16] **Unicode in BS2000/OSD**
Introduction

Target group
Application programmers and system administrators, who want to get an overview, to which extent the Unicode support is provided in BS2000/OSD, and which BS2000/OSD components you need for the Unicode support
Contents
This manual gives an overview of the Unicode support in BS2000/OSD and describes basics, concepts and correlations, which apply for all BS2000/OSD products concerned by Unicode. Thus it completes the product-specific description in the respective manuals. Helpful tables from the Unicode conversion surroundings complete the manual.

Index

Fujitsu Siemens Computers GmbH
User Documentation
81730 Munich
Germany

Fax: (++49) 700 / 372 00001

e-mail: manuals@fujitsu-siemens.com
http://manuals.fujitsu-siemens.com

Comments
Suggestions
Corrections

Submitted by

Comments on FHS V8.3A
Format Handling System for openUTM, TIAM, DCAM

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *…@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Copyright Fujitsu Technology Solutions, 2009

Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *…@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.

Copyright Fujitsu Technology Solutions, 2009