

FORMS

User's Guide

DPS7000/XTA
NOVASCAL 7000

Program Preparation



REFERENCE
47 A2 15UJ 05

DPS7000/XTA NOVASCALE 7000 FORMS User's Guide

Program Preparation

March 2000

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

REFERENCE
47 A2 15UJ 05

The following copyright notice protects this book under Copyright laws which prohibit such actions as, but not limited to, copying, distributing, modifying, and making derivative works.

Copyright © Bull SAS 1991, 2000

Printed in France

Suggestions and criticisms concerning the form, content, and presentation of this book are invited. A form is provided at the end of this book for this purpose.

To order additional copies of this book or other Bull Technical Publications, you are invited to use the Ordering Form also provided at the end of this book.

Trademarks and Acknowledgements

We acknowledge the right of proprietors of trademarks mentioned in this book.

Intel® and Itanium® are registered trademarks of Intel Corporation.

Windows® and Microsoft® software are registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark in the United States of America and other countries licensed exclusively through the Open Group.

Linux® is a registered trademark of Linus Torvalds.

The information in this document is subject to change without notice. Bull will not be liable for errors contained herein, or for incidental or consequential damages in connection with the use of this material.



Preface

Scope and Objectives	This manual describes the MAINTAIN_FORM utility that allows you to control the presentation of data on your terminal screen.	
Intended Readers	The intended readers of this manual are programmers who wish to create and/or maintain "forms", the control structures that determine how data is presented on a terminal's screen. A general knowledge of GCOS 7 (such as may be obtained by reading Part 1 of the <i>IOF Terminal User's Reference Manual</i>), is assumed, as well as familiarity with a programming language used for IOF programming (COBOL, C, FORTRAN, or GPL).	
Structure	Section 1	is an overview of the utility, including its structure, usage, and commands.
	Section 2	describes the commands used to create and modify forms.
	Section 3	describes the commands used to manage forms.
	Section 4	describes the format and usage of the Source Form Definition Language.
	Section 5	describes the operational features specific to various terminals.
	Section 6	contains examples of typical operations.
	Appendix A	describes the SDPI (Standard Device Programmatic Interface), which provides a means to handle forms from within a program.



Bibliography The following manuals are prerequisite reading to understand the material presented in this manual:

IOF Programmer's Manual 47 A2 37UJ
IOF Terminal User's Reference Manual - Part 1 47 A2 38UJ

The following manuals are also recommended:

IOF Terminal User's Reference Manual - Part 2 47 A2 39UJ
IOF Terminal User's Reference Manual - Part 3 47 A2 40UJ
IOF Terminal User's Reference Manual - Part 4 47 A2 34UJ
GCOS 7-V6 System Administrator's Manual..... 47 A2 10US
GCOS 7-V7/V8/V9 System Administrator's Manual 47 A2 54US

The following manuals should be consulted as needed:

COBOL 85 Reference Manual..... 47 A2 05UL
COBOL 85 User's Guide 47 A2 06UL
FORTRAN 77 Reference Manual 47 A2 15UL
FORTRAN 77 User Guide 47 A2 16UL
GPL Reference Manual 47 A2 35UL
GPL User's Guide..... 47 A2 36UL
GPL System Primitives Reference Manual..... 47 A2 34UL
C Language User's Guide..... 47 A2 60UL
C Language System Primitives Reference Manual..... 47 A2 64UL
TDS COBOL Programmer's Guide 47 A2 33UT



Syntax Notation	The following notation is used in the format of the commands presented in this manual:
ITEM	Uppercase characters are used for keyword items that are to be entered exactly as shown. Required words are underlined.
item	Lowercase characters are used for user-supplied items.
[item]	An item within square brackets is optional.
{item1 } {item2 } { <u>item3</u> }	A column of items within braces means one item must be selected. The default, if there is one, is underlined (e.g., item3 here).
{item1 item2 item3 }	A sequence of items separated by vertical bars has the same meaning as the convention just above.
...	An ellipsis indicates that the preceding item may be repeated one or more times.





Table of Contents

1. Overview

1.1	Introduction	1-1
1.2	MAINTAIN_FORM Organization	1-3
1.3	Language Members	1-5
1.3.1	COBOL Interface	1-5
1.3.1.1	Form Identification	1-5
1.3.1.2	Data Record	1-6
1.3.1.3	Selection Vector	1-7
1.3.2	C Interface	1-8
1.3.2.1	Form Identification Language Member	1-8
1.3.2.2	Data Record Language Member	1-10
1.3.2.3	Selection Vector Language Member	1-15
1.4	Calling MAINTAIN_FORM	1-18
1.5	List of Commands	1-20

2. Creating and Modifying Forms

2.1	ICREATE Command	2-1
2.1.1	Command Processing	2-3
2.1.1.1	Specification Phase	2-4
2.1.1.2	Drawing Phase	2-4
2.1.1.3	Description Phase	2-5
2.2	MODIFY Command	2-6
2.2.1	Command Processing	2-9
2.2.1.1	Specification Phase	2-9
2.2.1.2	Delete Phase	2-9
2.2.1.3	Drawing Phase	2-9
2.2.1.4	Description Phase	2-10
2.2.1.5	Methods of Modifying Forms	2-12
2.2.1.6	Warning	2-13
2.3	CREATE Command	2-14



2.4	EDIT Command	2-17
2.5	FSE Command.....	2-17
2.6	COMPILE Command	2-18
2.7	DECOMPILE Command	2-22
2.8	EXTRACT Command.....	2-23
3.	Managing Forms	
3.1	DELETE Command.....	3-1
3.2	DISPLAY Command	3-3
3.3	LIST Command	3-4
3.4	LOAD Command.....	3-6
3.5	MOVE Command	3-9
3.6	PRINT Command.....	3-11
3.7	QUIT Command	3-11
3.8	TEST Command	3-12
3.9	TRANSIT Command	3-12
4.	Source Form Definition Language	
4.1	General Format.....	4-1
4.1.1	Syntax Rules.....	4-2
4.1.2	General Rules.....	4-2
4.2	Pictorial Description	4-4
4.2.1	General Format.....	4-4
4.2.2	Syntax Rules.....	4-4
4.2.3	General Rules.....	4-4
4.3	UFIELD/NFIELD Statements	4-5
4.3.1	General Format.....	4-5
4.3.2	General Rules.....	4-5
4.3.3	Data-name Clause.....	4-6
4.3.4	Location Clause	4-6
4.3.5	LENGTH Clause	4-7
4.3.6	SCREEN-PICTURE Clause.....	4-7
4.3.7	VALUE Clause.....	4-11
4.3.8	USAGE Clause	4-11
4.3.9	FUNCTION Clause	4-12



4.4	ARRAY/END-ARRAY Statements	4-13
4.5	Attribute Codes	4-14

5. Terminal Dependent Features

5.1	DKU7005/7.....	5-1
5.1.1	Limitations.....	5-1
5.1.2	Attributes.....	5-2
5.1.3	RECEIVE REAL-MODE.....	5-2
5.1.4	Function Key Mapping.....	5-3
5.1.5	Switches	5-3
5.1.6	Local Form Storage on Diskette	5-3
5.1.7	Character Sets.....	5-3
5.2	DKU7107.....	5-4
5.2.1	Limitations.....	5-4
5.2.2	Attributes.....	5-4
5.2.3	RECEIVE REAL-MODE.....	5-5
5.2.4	Function Key Mapping.....	5-5
5.2.5	Technical Selections.....	5-5
5.2.6	Character sets	5-5
5.3	DKU7211.....	5-6
5.3.1	Limitations.....	5-6
5.3.2	Attributes.....	5-6
5.3.3	RECEIVE REAL-MODE.....	5-7
5.3.4	Function Key Mapping.....	5-7
5.3.5	Technical Selections.....	5-7
5.3.6	Character sets	5-8
5.4	VIP7804/HDS7/VIP8800.....	5-9
5.4.1	Limitations.....	5-9
5.4.2	Attributes.....	5-9
5.4.3	RECEIVE REAL-MODE.....	5-10
5.4.4	Function Key Mapping.....	5-10
5.4.5	72-Line Option	5-10
5.4.6	Character sets	5-10
5.5	PC7800	5-11
5.5.1	Limitations.....	5-11
5.5.2	Attributes.....	5-11
5.5.3	RECEIVE REAL-MODE.....	5-11



5.5.4	Function Key Mapping	5-12
5.5.5	Configuration	5-12
5.5.6	Use of the Keyboard	5-14
5.5.7	Character sets	5-14
5.6	IBM3270	5-15
5.6.1	Limitations	5-15
5.6.2	Attributes	5-15
5.6.3	RECEIVE REAL-MODE	5-15
5.6.4	Function Key Mapping	5-15
5.6.5	Character sets	5-15
5.7	IBM3278 - IBM3279	5-16
5.7.1	Supported Configuration	5-16
5.7.2	Limitations	5-16
5.7.3	Attributes	5-17
5.7.4	Color Attribute 3279 2B/3B	5-18
5.7.5	Character Sets	5-19
5.7.6	RECEIVE REAL-MODE	5-20
5.7.7	Function Key Mapping	5-20
5.7.8	Light Pen Processing	5-20
5.8	MINITEL (40 Columns)	5-21
5.8.1	Alphamosaic Character Set	5-21
5.8.2	Specific Features	5-23
5.8.3	Limitations	5-23
5.8.4	Attributes	5-24
5.8.5	RECEIVE REAL-MODE	5-24
5.8.6	Function Keys	5-24
5.8.7	Initial Values	5-25
5.9	MINITEL 1B (Mixed Mode 40/80 Columns)	5-26
5.9.1	Switching between 40 column and 80 column mode.	5-26
5.9.2	40 Column Mode	5-26
5.9.3	80 Column Mode	5-26
5.9.4	Limitations	5-26



6. Examples Using Menus

6.1	ICREATE Examples.....	6-1
6.1.1	ICREATE Menu: A Form Named SALES is to be Defined.....	6-1
6.1.2	Specification Phase	6-2
6.1.3	Drawing Phase	6-2
6.1.4	ENTER NF Request	6-3
6.1.5	Attribute Definition Form.....	6-4
6.1.6	ENTER UF Request	6-5
6.1.7	ENTER ARRAY Request.....	6-6
6.1.8	RETRIEVE Requested	6-7
6.1.9	RETRIEVE Request	6-8
6.1.10	TEST Request	6-9
6.1.11	QUIT Request.....	6-10
6.1.12	Menu of the TEST Command.....	6-11
6.1.13	Result of the TEST Command.....	6-12
6.1.14	The MODIFY is Prompted	6-13
6.1.15	Specification Phase	6-14
6.1.16	Drawing Phase	6-15
6.1.17	Drawing Phase - Modification.....	6-16
6.1.18	Description Phase.....	6-17
6.1.19	Description Phase - Error	6-18
6.1.20	Description Phase - Array Definition Request	6-19
6.1.21	Description Phase - Array Definition.....	6-20
6.1.22	Description Phase - RETRIEVE Request.....	6-21
6.1.23	Description Phase - Array definition	6-22
6.1.24	Description Phase - NF Request.....	6-23
6.1.25	Description Phase - RETRIEVE Request.....	6-24
6.1.26	Description Phase - RETRIEVE Request Result	6-25
6.1.27	Description Phase - New Location	6-26
6.1.28	Description Phase - RETRIEVE Request.....	6-27
6.1.29	Description Phase - RETRIEVE Request Result	6-28
6.1.30	Description Phase - New Location	6-29
6.1.31	Description Phase - TEST Request.....	6-30
6.1.32	Description Phase - TEST Request Result	6-31
6.1.33	End of Description Phase	6-32
6.1.34	Save Prompt	6-33



A. SDPI Interface

A.1	Form Generation	A-1
A.2	Principles of the SDPI Interface	A-3
A.3	Form Activation	A-4
A.4	Output Actions.....	A-6
A.5	Input Actions	A-6
A.6	Traces	A-7
A.7	SDPI COBOL Statements.....	A-8
A.7.1	Data Structures.....	A-8
A.7.2	CSEND (Forms Send).....	A-10
A.7.3	CDRECV (Forms Receive).....	A-12
A.7.4	CDGET (Form Activation).....	A-15
A.7.5	CDRELS (Release All Forms)	A-18
A.7.6	CDPURGE (Purge Input Data)	A-20
A.7.7	CDATTR (Attribute Selection)	A-21
A.7.8	CDATTL (Attribute List Selection)	A-25
A.7.9	CDFIDI (Form Identification).....	A-28
A.7.10	CDMECH (Control Function Mechanism).....	A-29
A.8	A Programming Example	A-32

Index



Table of Graphics

Figures

- 1-1. FORMS Facility Block Diagram 1-4
- 1-2. Member Storage and Related Command 1-22

Tables

- 5-1. Alphamosaic Character Set (1/2)..... 5-21





1. Overview

1.1 Introduction

MAINTAIN_FORM (MNFORM) is a utility that allows you to create and manage forms. The SDPI interface (described in Appendix A) enables you to use the forms created either via TPRs (under TDS) or via interactive programs (under IOF).

Forms are a convenient way of handling dialogue with a terminal. As both terminal and presentation characteristics are specified in the form, the program itself becomes terminal independent.

If you do not use forms, the control sequences to handle the terminal dialogue must be included in the program itself. These sequences are dependent on the terminal type, so a different sequence must be given for each terminal type; consequently, the program becomes unwieldy. In contrast, a program using forms treats each terminal as a symbolic terminal (with terminal-dependent characteristics specified via the form) and such a program can easily be run concurrently on terminals of different types.

Using forms also enables you to make a program independent of its presentation. A form represents a portion of a screen. It is composed of fixed text which is never modified and variable parts that may be modified. The modification can be done either by the program or by the terminal user, or by both.

The definition of the fixed text and the location (on the screen) of the variable parts are defined in the form, not in the program. The program sees the form as a data record whose fields contain the variable parts of the screen. You can use MAINTAIN_FORM to change the fixed texts (or the location of the variable parts) of the form without the program being affected in any way. Such changes do not necessitate a re-compilation of the program.



A form is defined as a set of fields on the screen. There are two types of field: named and unnamed. An unnamed field is not visible to the application program which calls the form. The contents of an unnamed field is fixed and is never transmitted to or from the application program. The contents of a named field can be altered by the user at the terminal and/or can be received or sent by the application program. The application program sees only named fields, each of which has a unique name.

An array can be created in a form by duplicating a line (or set of contiguous lines) several times. A form can have up to 255 named fields including the repeated fields in an array and the implicit function key fields. There may be other limitations, depending on the type of terminal used. Details of terminal types which can access forms created by MAINTAIN_FORM, together with their limitations and peculiarities, is given in the chapter "Terminal Dependent Features" later in this guide.

Access to the forms created by MAINTAIN_FORM is provided in COBOL, C, and GPL under IOF and TDS. The programmatic interface for forms is described in the *IOF Programmer's Manual*.

The MAINTAIN_FORM utility is available under both IOF and batch. However, you are recommended to use it under IOF to profit from its interactive facilities.



1.2 MAINTAIN_FORM Organization

MAINTAIN_FORM takes its input from:

- The IOF terminal from which MAINTAIN_FORM is submitted for execution. Here a sequence of commands previously stored in a source library may be submitted to MAINTAIN_FORM using the ALTER INPUT directive.
- An input enclosure or a library member (H_CR) in BATCH.

The objects generated by MAINTAIN_FORM are stored in a binary library (BINLIB), in a source library (SLLIB), and optionally in a UFAS indexed file (OUTFILE) or on a diskette (DK).

For each form, MAINTAIN_FORM generates:

- A reference source form member in BINLIB, whose name is formname_SF.
- A terminal-independent object member (formname_of_ANY) or as many object members as there are terminal types for which the form has been loaded (formname of terminaltype). These members are created in the binary library (BINLIB).
- Language members in SLLIB, to be retrieved by user programs through COPY statements or equivalents (include member for C language, macro-instruction for GPL). The contents of these members are detailed under "Language Members", below.

Moreover, MAINTAIN_FORM provides commands to:

- Copy a reference source form member from BINLIB to SLLIB for modification (DECOMPILE). This member will have the same name as the form it defines.
- Move an object form from BINLIB to OUTFILE or to a terminal diskette. The OUTFILE may be subsequently used to retrieve object forms under TDS, in order to get better performances. Each object form member is moved as 2 records in OUTFILE known as '.F' (object form) and '.M' (mapping information).

The complete organization of the FORMS facility is described in Figure 1-1.

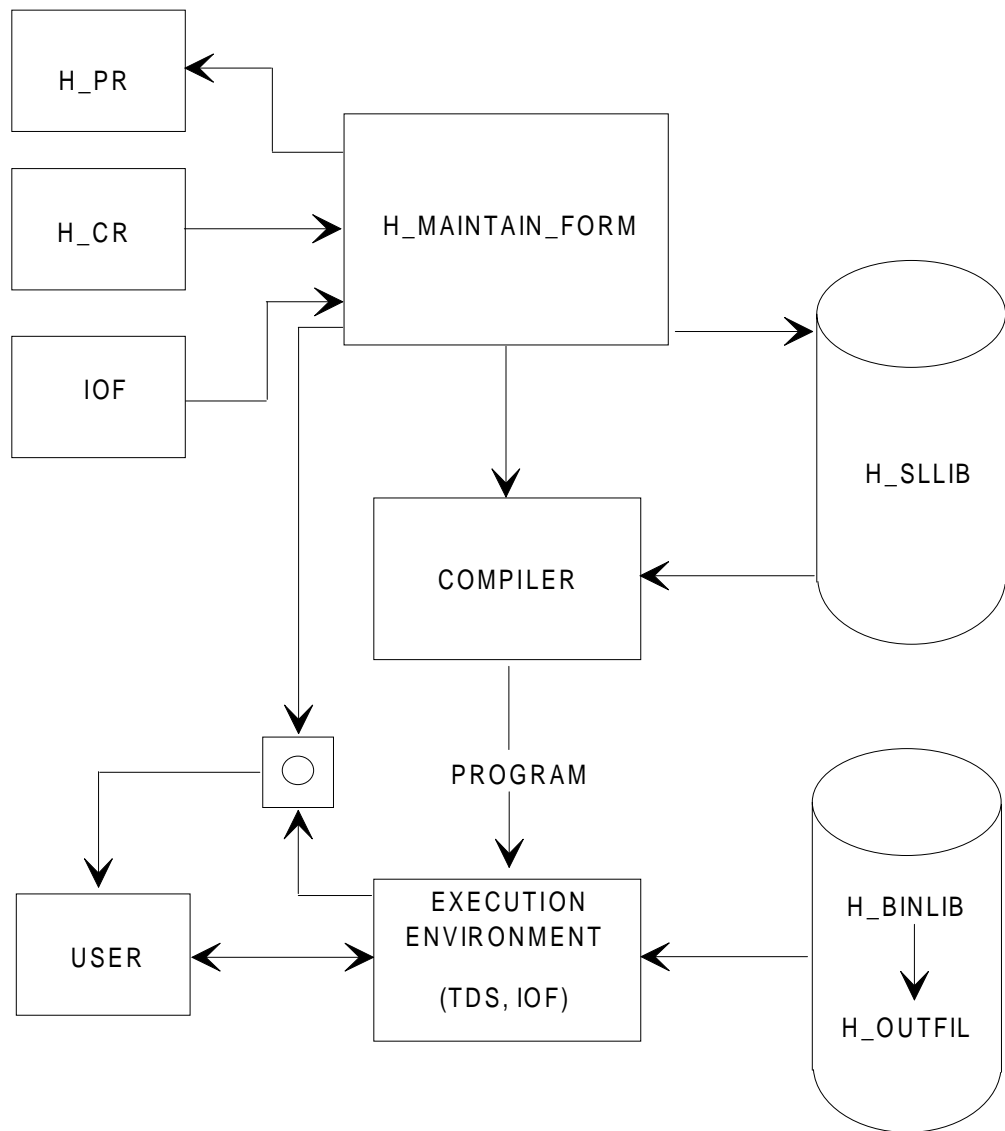


Figure 1-1. FORMS Facility Block Diagram



1.3 Language Members

This subsection discusses the contents of language members generated by MAINTAIN_FORM. The use of the data items contained in these members is described under the heading "Forms Mode: SDPI Interface" in the *IOF Programmer's Manual*.

MAINTAIN_FORM usually generates three language members: form identification, data record, and selection vector.

When a form is to be used under the IQS relational information system, no language member is generated in SLLIB but an additional data description member is generated in BINLIB. Note that if a form is to be used with IQS, you must specify LANG=DD at creation time.

1.3.1 COBOL Interface

1.3.1.1 Form Identification

This structure is used to uniquely identify a form, to check its compilation date and to specify its mode of activation.

```
01 data-name.  
COPY formnameI
```

Structure:

```
*  
  
02 FILLER PIC X VALUE "3".  
02 FILLER PIC X(8) VALUE "form-name".  
02 formname-NO PIC 9(3) VALUE ZERO..  
02 formname-MD PIC X VALUE "A".  
02 formname-OF PIC X(8) VALUE SPACES.  
02 formname-OO PIC 9(3) VALUE ZERO.  
02 formname-LL PIC 9(3) VALUE ZERO.  
02 FILLER PIC X(5) VALUE "date-of-compilation".  
02 FILLER COMP-1 VALUE time-of-compilation.  
02 formname-AF PIC 9 VALUE any-flag.  
02 formname-SL PIC 9(3) VALUE 1.  
02 formname-SC PIC 9(3) VALUE 1.
```



Where:

- formname-NO is used to specify the new occurrence number of the form.
- formname-MD is the mode of activation of the form (A stands for APPEND, O for OVERLAY, W for WINDOW and E for ERASE).
- formname-OF is the name of the old form to which the new form is to be appended or overlaid.
- formname-OO is the occurrence number of this old form.
- if formname-LL is zero, the number of lines allocated is equal to the number of lines in the form. If formname-LL is greater than or equal to the number of lines in the form, the form is completed with the appropriate number of blank lines. If formname-LL is less than the number of lines in the form (but greater than zero), the status FUNCNAV is returned.
- formname-AF is the "any-flag" which is initialized according to the TERM option selected at form-loading (1 if ANY, 0 else).
- formname-SL is the form starting line in case of W and E modes.
- formname-SC is the form starting column in the case of W and E modes.

1.3.1.2 Data Record

This structure is used to send and receive the fields of the form.

```
01 data-name.  
COPY formnameR
```

Structure:

```
04 formname-FC PIC 99.  
04 dataname-01 PIC ...  
04 dataname-02 PIC ...  
...  
04 arrayname-A.  
05 arrayname OCCURS dimension.  
06 dataname-i PIC ...
```



The first field of the data record is generated by default, except when the NO IMPLICIT FUNCTION-CODE clause is selected or an explicit FUNCTION-CODE field is specified. This field is used to receive the function code as a terminal-independent rank. Named field entries are generated following this field for each NF statement. The PIC clause of an entry is derived from the SPIC clause of the corresponding NF statement as specified in Section 4, Source Form Definition Language. The order of these entries in the record is the same as the order of NF statements in the source form definition.

1.3.1.3 Selection Vector

This structure is used to select individual fields of a form for the purpose of sending or receiving their contents, to modify their attributes or to return the status of a field.

```
01 dataname.  
COPY formnameV
```

Structure:

```
02 formnameV.  
  03 FILLER PIC X VALUE ""2".  
  03 FILLER COMP-1 VALUE number-of-fields.  
  03 FILLER PIC X(8) VALUE "form-name".  
  03 formname-VO PIC 9(3) VALUE ZERO.  
  03 formname-V.  
    04 formname-FC-V PIC X.  
    04 dataname-01-V PIC X.  
    04 dataname-02-V PIC X.  
  ...  
  04 arrayname-AV.  
    05 arrayname-V OCCURS dimension.  
    06 dataname-i-V PIC X.
```

The formname-VO field is the occurrence number of the form. The formname-FC-V field is the selection vector entry corresponding to the formname-FC field in the data record. The order of data-names is the same as the order of data names in the data record.



1.3.2 C Interface

1.3.2.1 Form Identification Language Member

The type of a form identification structure is named `struct FORMNAMEi` and is defined in the `FORMNAMEI_H` source member generated by `MAINTAIN_FORM`. An initialization function of the form identification must be provided by `FORMNAMEI_H`.

A structure of this type is used to uniquely identify a form, to check its compilation date, and to specify its mode of activation.

Structure Definition

The structure type synopsis is defined as follows:

```
#include <FORMNAMEi.h>

struct FORMNAMEi {
char unused;
char formnm[8];
char formnm[8];
char no[3];
char md;
char of[8];
char oo[3];
char ll[3];
char date[5];
short time;
char af;
char sl[3];
char sc[3];
};
```

where:

- `formnm` specifies the form name. It is right padded with blanks, and there is no terminating null character.
- `no` specifies the new occurrence number of the form.
- `md` is the form activation mode, using the following values:

A stands for APPEND
O stands for OVERLAY
W stands for WINDOW
E stands for ERASE



- `of` is the name of the old form that the new form appends. It is right padded with blanks, and there is no terminating null character.
- `oo` is the occurrence number of the old form.
- `ll` is line data. If `ll` is 0, the number of lines allocated is equal to the number of lines in the form. If `ll` is greater than or equal to the number of lines in the form, the form is completed with the appropriate number of blank lines. If `ll` is less than the number of lines in the form, but greater than 0, the return code `FUNCNAV` is returned.
- `date` and `time` specify the date and time of the form compilation.
- `af` is any-flag. The `TERM` option defines the initialization of this field. The `TERM` option is selected at form loading time 1 if ANY, otherwise 0).
- `sl` and `sc` determine the form starting line and column in the case of `WINDOW` and `ERASE` modes.

Structure Initialization

Before the very first use of a structure whose type is `struct FORMNAMEi`, it should be initialized with the `FORMNAMEi` function. It is defined as follows:

```
#include <FORMNAMEi.h>

void h_init_FORMNAMEi( form_i )
struct FORMNAMEi form_i;
```

Using `h_init_FORMNAMEi` is equivalent to the following sequence:

```
form_i.unused = '3';
cpybuf(form_i.formnm, "FORMNAME", 8);
cpybuf(form_i.no, "000", 3);
form_i.md='A';
cpybuf(form_i.of, "          ", 8);
cpybuf(form_i.oo, "000", 3);
cpybuf(form_i.ll, "000", 3);
cpybuf(form_i.date, "yyddd", 5);
form_i.time = 100 * ( hh + mn/60 );
form_i.af = flag;           /* flag is '1' or '0' */
cpybuf(form_i.sl, "001", 3);
cpybuf(form_i.sc, "001", 3);
```



The syntax to be generated by `MAINTAIN_FORM` inside the header file is:

```
#define h_init_FORMNAMEi(_frm)\
cpybuf((char *)&(_frm),\
"3FORMNAME000A      000000yyddd\hhh\hhhf001001",\
sizeof(struct FORMNAMEi))
```

where:

- `yyddd` are 5 numeric characters. Example: 89363 if the form was compiled on December 29, 1989.
- `\hhh\hhh` is the octal encoding on two bytes of the compilation time. Example: `\005\303` for the representation of the decimal value 1475 (i.e.; 14h45).
- `f` is either 1 or 0 according to the "any-flag" value.

NOTE:

The advantage of this type of definition is to expand the function call into only one expression.

Any `#define` (preprocessor substitution macro) concerns only the line it begins. If the substituting expression is too long to be written on a single line, any end-of-line should be preceded by a backslash `\`.

The character `#` of `#define` or of any other preprocessor commands must be the first one of the line.

1.3.2.2 Data Record Language Member

The type of a data record structure is named `struct FORMNAMEr`, and the `FORMNAMEr_H` source member defines it. `MAINTAIN_FORM` generates the `FORMNAMEr_H` source member.

This structure type contains the declarations in C language of the form's named fields (NF). It sends and receives the fields of the form.



The structure type synopsis is as follows:

```
#include <FORMNAMEr_h>

struct FORMNAMEr {
char fc[2];
C_type DATA_NAME_01;
C_type DATA_NAME_02;
...
struct {
    C_type DATA_NAME_i;
    ...
} ARRAY_NAME [nb-dim];
...
};
```

The first field of the data record (`fc`) is generated by default, except when the `NO IMPLICIT FUNCTION CODE` clause is selected when the form is created. This field receives the function code as a terminal-independent rank. Named field entries are generated following this field for each `NF` statement in the source form definition. The order of these entries in the record is the same as the order of `NF` statements in the source form definition.

Relationship Between Pictorial Description and Data Record

The fields of a data record are related to the named fields of the pictorial description of a given form. Each named field is defined in the Source Form Definition Language of `MAINTAIN_FORM` as follows:

```
{ NFIELD | NF } [data-name] [location] [length]
[ [usage] [] ] .
```

For more details about syntax and the creation of forms, see Section 4. The paragraphs below summarize the information specific to the support of C language syntax.



Data Name Clause

Data-name is the name given to the associated data record field, with the restriction that hyphens of the source form definition be replaced by underscores in the C language data record structure.

For example,

```
NF DATA-01 LENGTH IS 2.
```

generates the structure member:

```
char DATA_01[2];
```

If the data name clause is omitted, the standard name *FORMNAME_nnn* will be used, where *nnn* stands for the rank of the named field in the data record and ranges from 01 to 255.

Length Clause

The length clause syntax is as follows:

```
{ LENGTH | LEN } IS integer
```

If the length clause is defined without the usage clause, it gives the size in bytes of the data record associated structure member.

For example,

```
NF DATA-02 LENGTH IS n .
```

generates either:

```
char DATA_02;      /* when n=1/
```

or

```
char DATA_02[n];  /* when n>1/
```

Picture Clause

The picture clause syntax is as follows:

```
{ SCREEN-PICTURE | SCREEN-PIC } IS character-string
```



The picture clause is taken into account by the data record when no usage clause is specified. Three categories of data can be described with a picture clause:

- Alphanumeric data

Character-string is restricted to the symbol X. For example, XXX, as well as X(3), describes up to 3 alphanumeric characters.

```
NF DATA-03 SCREEN-PICTURE IS X(n) .
```

generates either:

```
char DATA_03;          /* when n=1 */
```

or:

```
char DATA_03[n];      /* when n>1
```

- Numeric data

Character-string is restricted to the symbol 9. For example, 999, as well as 9(3), describes up to 3 numeric characters.

```
NF DATA-04 SCREEN-PICTURE IS 9(n) .
```

generates either:

```
char DATA_04;        /* when n=1 */
```

or:

```
char DATA_04[n];    /* when n>1 */
```

- Numeric edited data

Character-string is restricted to certain combinations of the symbols 9 Z * + - . (or ,) and the currency sign. The allowable combinations are described in Section 4. However, it should be noted that each occurrence of 9, Z, or * is counted in the size of the data record generated item.. Furthermore, an integer that is enclosed in parentheses following the symbol 9, Z, or * indicates the number of consecutive occurrences of the symbol. Assuming that \$ is the currency sign, the following examples:

```
NF DATA-05 SCREEN-PICTURE IS ZZ99 .  
NF DATA-06 SCREEN-PICTURE IS **9.99 .  
NF DATA-05 SCREEN-PICTURE IS +9(x).9(y) .  
NF DATA-05 SCREEN-PICTURE IS $Z(x)9.9(y) .
```

respectively generate:

```
char DATA_05[4];  
char DATA_06[5];  
char DATA_07[x+y];  
char DATA_08[x+1+y];
```



Usage clause

The usage clause syntax is as follows:

```
USAGE IS { COMP-1 | COMP-2 | COMP-8 }
```

A usage clause implies a numeric or numeric edited picture. For COMP-1 and COMP-2, the picture clause must not contain a decimal point.

COMP-1 usage is equivalent to a `short` integer (C language) or `FIXED BIN(15)` (GPL) type. COMP-2 usage is equivalent to a `long` integer (C language) or `FIXED BIN(31)` (GPL) type.

For example:

```
NF DATA-09 SCREEN-PICTURE IS 9(x) USAGE IS COMP-1 .  
NF DATA-10 LENGTH IS y USAGE IS COMP-2 .
```

respectively generate:

```
short DATA_09;  
short DATA_10;
```

The use of COMP-3 or COMP-8 is not allowed when the generation of C language data records is requested. If these types are used a severity 2 message is issued:

```
**PACKED PICTURE NOT ALLOWED IN C LANGUAGE
```

and the associated NF statement is ignored.

Array Statements

The array statements of a source form definition are used to define a repetitive group of fields in a form. The general format of an array definition is as follows:

```
ARRAY data-name OCCURS integer .  
{ NFIELD/UFIELD statement } ...  
END-ARRAY .
```

UFIELD (i.e., unnamed fields) statements that are part of the array are not concerned with the data record generation.



The following example:

```
ARRAY PRICE-LINES OCCURS n.  
NF US-PRICE LINE 04 COL IS 10 SCREEN-PIC IS ZZ9.99 NU .  
UF LINE IS 04 COL IS 17 VALUE IS "$" .  
NF UK-PRICE LINE 04 COL IS 30 SCREEN-PIC IS ZZ9.99 NU .  
UF LINE IS 04 COL IS 37 VALUE IS "£" .  
END-ARRAY.
```

generates:

```
struct {  
    char US_PRICE[5];  
    char UK_PRICE[5];  
} PRICE_LINE[n];
```

1.3.2.3 Selection Vector Language Member

The type of a selection vector structure is named `struct FORMNAMEv` and is defined in the `FORMNAMEV_H` source member that `MAINTAIN_FORM` generates. `FORMNAMEV_H` also provides an initialization function of the form identification.

This structure type selects individual fields of a form to send or receive their contents, modify their attributes or return the status of a field.



Structure Definition

The structure type synopsis is defined as follows:

```
#include <FORMNAMEv_h>

struct FORMNAMEv {
char unused;
short nbfield;
char formnm[8];
char vo[3];
struct {
char fc;
char DATA_NAME_01;
char DATA_NAME_02;
...
struct {
char DATA_NAME_i;
...
} ARRAY_NAME [nb-dim];
...
} vector;
};
```

where:

- `vo` is the occurrence number of the form.
- `vector.fc` is the selection vector entry corresponding to the `fc` field in the data record (struct *FORMNAMEv* type).
- `vector` contains one declaration `char` for each of the `nbfield` fields of the form. The order of data-names is the same as the order of data-names in the data record.

Structure Initialization

Before the very first use of a structure whose type is struct *FORMNAMEv*, it should be initialized with the `h_init_FORMNAMEv` function. It is defined as follows:

```
#include <FORMNAMEv_h>

void h_init_FORMNAMEv( form_v )
struct FORMNAMEv form_v;
```




Using `h_init_FORMNAMEv` is equivalent to the following sequence:

```
form_v.unused = 0x01;
form_v.nbfield = nbfield;
cpybuf (form_v.formnm, "FORMNAME", 8);
cpybuf(form_v.vo, "000", 3);
```

The syntax to be generated by `MAINTAIN_FORM` inside the header file is:

```
#define h_init_FORMNAMEv(_frm)\
cpybuf((char *)&(_frm), \
"\001\nnn\nnnFORMNAME000", \
sizeof(struct FORMNAMEv)-sizeof((_frm),vector))
```

`\nnn\nnn` is the octal encoding of the number of fields *nbfield* of the form. For example, if the form contains 39 named fields, the octal representation is `\000\047`.



1.4 Calling MAINTAIN_FORM

MAINTAIN_FORM can be called in either IOF or Batch mode.

In IOF, it is called through the GCL command:

```
MAINTAIN_FORM  
[BINLIB=binary-library] [SLLIB=source-library]  
[OUTFILE=file-description [FSHARE={ONEWRITE|NORMAL|MONITOR}]];
```

If BINLIB is not specified, MAINTAIN_FORM uses either the default binary output library of where:the session (BLIB), if present, or TEMP.BILIB\$TEMPRY.

IF SLLIB is not specified, MAINTAIN_FORM uses either the default source output library of terminating null character.the session (SLIB), if present, or TEMP.SLLIB\$TEMPRY.

OUTFILE must be specified when a command referencing the UFAS output file is to be submitted. It must have been allocated with the following parameters:

```
RECSIZE   : 6100  
CISIZE    : 12288  
KEYSIZE   : 20  
KEYLOC    : 1  
RECFORM   : V
```

Note that giving smaller values to RECSIZE and CISIZE is acceptable as long as the forms to be stored in the file are small enough to fit with these values.

The default sharing option of OUTFILE is ONEWRITE (with ACCESS = WRITE). This option is advised for the following reasons:

- This sharing allows one MAINTAIN_FORM user to actually share the file with a TDS having the file assigned with SHARE = ONEWRITE and ACCESS = READ, which is not the case if SHARE = NORMAL is used.
- There is no concept of commitment within MAINTAIN_FORM, which implies that using SHARE = MONITOR may roll back the entire MAINTAIN_FORM session.

Consequently, if several users want to create forms simultaneously and test them under TDS, they should directly assign a binary library to the TDS step with SHARE = DIR. The use of OUTFILE is intended to get better performances and it is recommended that this use be reserved for a System Administrator.



In Batch, MAINTAIN_FORM is called through the following basic JCL:

```
STEP H_FORMGEN, FILE= SYS.HLMLIB;  
ASSIGN H_BINLIB, library-description;  
ASSIGN H_SLLIB, library-description;  
ASSIGN H_CR,input-enclosure;  
    [ASSIGN H_OUTFIL, file-description;]  
ENDSTEP;
```



1.5 List of Commands

The following commands are available under MAINTAIN_FORM. Commands with a star are available to IOF only from a terminal supported by the FORMS run-time package. For each command, menus and Help texts are provided through prompts. The name of each command is followed by its abbreviation.

Commands for creating and modifying forms:

*ICREATE, ICR	Create a new form in full-screen mode
*MODIFY, MD	Update the definition of a form in full-screen mode
CREATE, CR	Create a new form from definition statements
EDIT, ED	Use the Text Editor on SLLIB
* FSE	Use the Full Screen Editor on SLLIB
COMPILE, COMP	Compile a source form from SLLIB
DECOMPILE, DEC	Decompile a source form from BINLIB to SLLIB
EXTRACT, EXTR	Extract an object form from a member which was transferred from a VIDEOTEX diskette

Commands for form management:

DELETE, DL	Delete a form
DISPLAY, D	Display an image of a form
LIST, LS	List form names
LOAD, LD	Load object forms and language members
MOVE, MV	Move object forms
PRINT, PR	Print a source form from BINLIB
QUIT, Q	Leave MAINTAIN_FORM
TEST*	Test an object form in BINLIB
TRANSIT, TRAN	Transit from a 1E SLLIB source to BINLIB



NOTE:

There are two methods of creating and modifying forms. The recommended way is to use the screen prompts on the terminal for ICREATE and MODIFY commands. The form may be tested at any time during the creation or modification.

The other way is to use the source form definition language, in which the CREATE command defines forms. To modify a form, the DECOMPILE command creates a source member in the source library, and the Text Editor (FSE and EDIT commands) modifies it. The form is then regenerated by the COMPILE command.

The specification of the source form definition language is given under the heading "Source Form Definition Language" in this section. If you use only the ICREATE and MODIFY commands, you need not become familiar with this language. However, a knowledge of this language enables you to understand the parameter values used at form and field level to define a form.

Since when working in full-screen mode, MAINTAIN_FORM generates source definition statements automatically, which may then be retrieved by DECOMPILE, it can be said that both ways of managing forms can be combined.

The commands of the MAINTAIN_FORM processor and their relationships with the different types of member are shown in Figure 1-2 below.

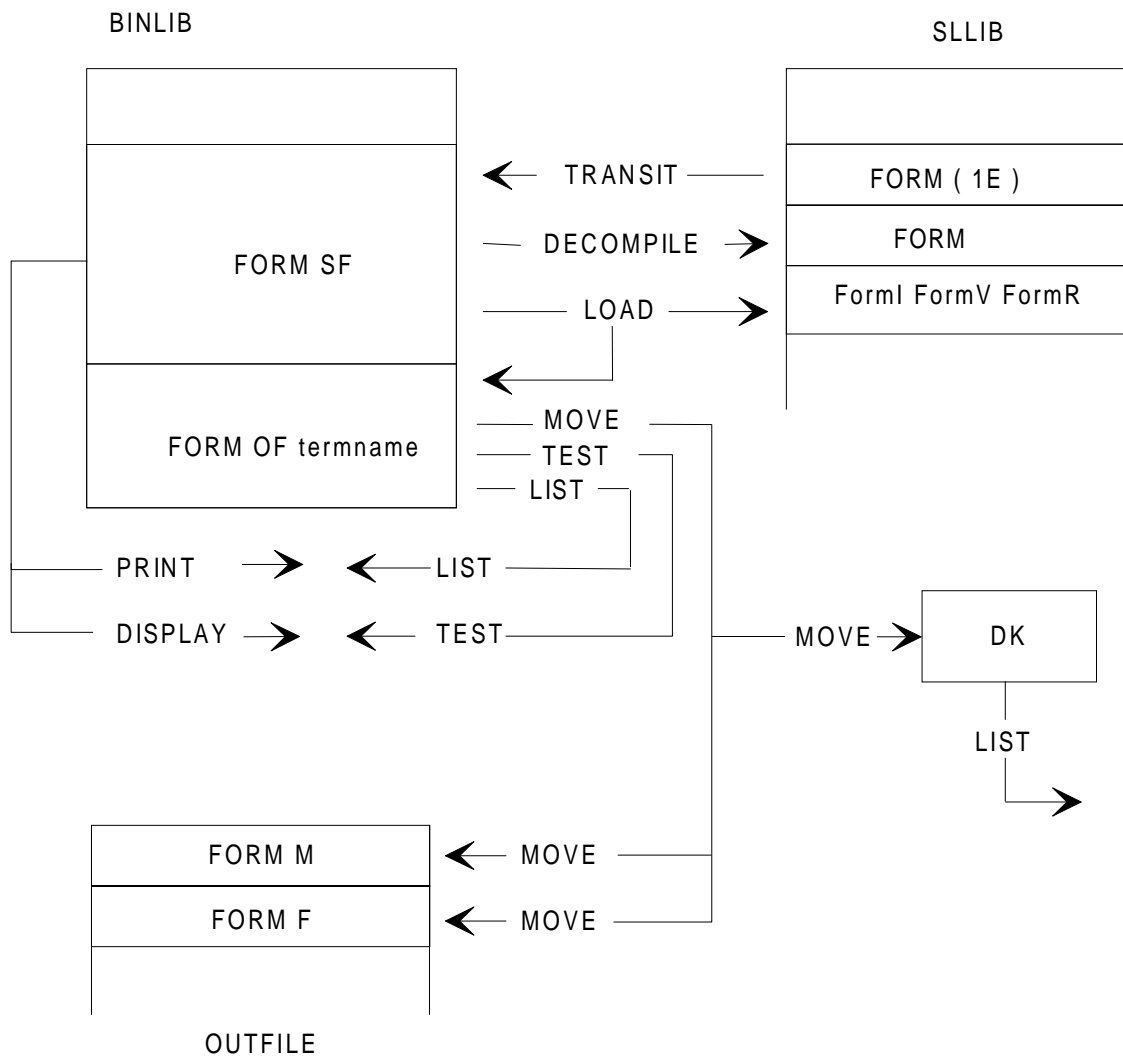


Figure 1-2. Member Storage and Related Command



2. Creating and Modifying Forms

2.1 ICREATE Command

Syntax:

```
ICREATE form-name  
  
          { terminal-type [,terminal-type ] ... }  
[TERM = { ANY                } ]  
          { ALL                }  
  
[LANG={COBOL DD CL GPL}]  
  
[RECORD=record-name];
```

where terminal-type =

```
{DKU7007|DKU7005|IBM3270|VIP7804|DKU7107|DKU7211|MINITEL|  
PC7800|IBM3278|IBM3279}
```

Purpose:

This command allows you to create a new form on the screen under IOF. Help texts can be requested at any level during the creation of this form, and the form may be continuously tested while it is being created. Although this command does not use the form definition language, it can access most options available in that language. Options which are provided for compatibility reasons (e.g., RECEIVE REAL-MODE) cannot be selected, but may be added later through either FSE or EDIT commands. They will remain unmodified in the case of subsequent use of the MODIFY command. The order of the fields in the data record will be the same as the order of the fields on the screen.

**Parameters:**

form-name	The name, up to 8 characters long, of the form to be created. The first character must be alphabetic and the others alphanumeric - the minus character is allowed.
TERM	<p>The terminal type or list of terminal types (separated by commas) for which the object form must be loaded. If ANY is specified, instead of generating a terminal-dependent member, MNFORM generates a terminal-independent member which is adapted to the terminal at run-time. This terminal-independent object member cannot be moved to the UFAS outfile or to the terminal diskette. If ALL is specified, the form is loaded for all terminal types that are supported.</p> <p>If no terminal type is specified, the default is the type of terminal on which the user is logged.</p> <p>MNFORM stores a flag in the form-identification structures that indicates whether or not the ANY option has been selected. When the form is accessed at run-time and if ANY was not selected, FORMS looks in each binary library, first for the terminal-dependent member then, if not found, for the terminal-independent member. If ANY was selected, FORMS looks first for the terminal-independent member, then for the terminal-dependent member.</p> <p>Terminal type IBM3278 is mapped as IBM3279.</p> <p>When several terminal types exist for a given form, it is possible to modify the form for one a single terminal type only. But this operation may lead to an incoherence among the terminal type members which are formname-of-terminal-type.</p> <p>For instance if formname-of-DKU7007 and formname-of-IBM3270 both exist, and if formname-of-DKU7007 is modified, the members formname-of-DKU7007 and formname-of-IBM3270 become different.</p> <p>We recommend that the form be modified for all existing terminal-types.</p> <p>This recommendation is also valid when ANY-terminal-type and other terminal-types simultaneously exist for a given form.</p>



LANG (alias TYPE). The language in which language members are to be loaded. The default is the value of the #FORMLANG system variable. The acceptable values of #FORMLANG are COBOL, CL, and GPL; the default is COBOL. You can modify the value of the #FORMLANG variable by using the GCL directive MODIFY_PROFILE; e.g.:

```
MODIFY_PROFILE  
FORMLANG=CL ;
```

When DD is specified, a data description member is generated in BINLIB. No language members are generated in SLLIB. DD should be used only for forms to be used with IQS.

RECORD This is meaningful only when LANG=DD. It indicates the name of the record to be retrieved by IQS. When RECORD is not specified, the form name is taken as the record name. The name of the data description member is record-name D.

2.1.1 Command Processing

Form creation consists of three phases:

- a specification phase
- a drawing phase
- a description phase.

Help texts are available for the specification and description phases.



2.1.1.1 Specification Phase

You receive a form which allows you to specify the parameters below:

- Currency sign (default is \$)
- Decimal point representation (period or comma)
- Description (optional text that will generate a comment in the form identification record).
- Named field character: the character to be used in the next phase to denote named fields.
- Variable field ending character: the character to be automatically generated at the end of a variable field on a VIP 7700/7001. (This is specific to this type of terminal where a variable field must be terminated by a non space character).
- Unnamed field starting and ending characters: the characters to be used in the drawing phase to enclose the UFIELDS for which rendition attributes will be specified in the third phase.
- Receive space: when this option is selected, empty fields and fields filled with spaces will be considered as "received" at run time. If this option is not selected (the default), then such fields are considered as "not received" at run time.
- ALLOW RPLF: when this option is selected, the arrays defined by the terminal operator during the description phase are generated with RPLF when the form is generated for a DKU terminal family. If the ALLOW RPLF option is not selected (the default), this terminal functionality is not used.
- Receive function key: when this option is selected (the default), an implicit function key field is generated. When this option is not selected, an implicit function key is not generated. This option is forced if no named fields are declared.

2.1.1.2 Drawing Phase

The screen is cleared so that you can draw an image of the form using the conventions specified in the preceding phase. In this image, the unnamed fields are to be input as they will appear at run time. Moreover, those for which a rendition attribute will be specified must be enclosed with the appropriate delimiting characters. When a form is created from a terminal with pseudo graphic capabilities, unnamed fields with the pseudo graphic character set attribute may be generated by directly entering pseudo graphic characters from the keyboard. The named field locations are to be filled with named field characters, as specified in the preceding phase.

When an array is to be generated, only the first entry of the array is to be drawn. This entry should consist of one or several lines and followed by as many lines as necessary to expand the array. The lines where the array is to be expanded must not contain named fields; they may contain unnamed fields provided that these fields will not overlap with the fields generated by the expansion.



2.1.1.3 Description Phase

In this phase, field parameters supplementary to those drawn in the drawing phase, can be added. Arrays can also be expanded.

Now the screen is divided into two parts. In the upper part, a window on the form drawing is displayed. In the lower part, you are prompted with one of three possible forms: NF definition form, UF definition form or ARRAY definition form. These forms enable you to control the window by specifying the first line to be displayed and/or to specify a request.

The following requests are available:

- **ENTER NF,UF, or ARRAY:** When the form of the lower part of the screen does not match the type of item to be entered, the result of this request is to replace this form by the appropriate one. Otherwise, this request enters the parameters specified in the form in the description of the referenced item. After the execution of the request these parameters are cleared on the screen.
- An NF or UF is referenced by its line number and its rank within the line. For both NF and UF, field attributes may be specified.
- Up to three attributes may be directly entered in the corresponding attribute fields of the NF and UF forms, using the standard keywords. To enter more than three attributes, or to be able to enter an attribute without knowing the corresponding keyword, you must enter a question mark in the first attribute field. Then you will be prompted with an attribute definition form. In case of an NF, you may also specify the field name, its screen picture and its initial value. The screen picture controls editing functions to be performed on the field before it is displayed as well as valid data input. A detailed specification of screen pictures is given under the heading "SCREEN-PICTURE Clause" below. The NFs for which no name is specified will be automatically given a name at the end of the creation process. An ENTER NF (or UF) overrides the previous ENTER NF (or UF) for the same field. An ENTER NF (or UF) is executed only if a significative field (other than line and rank) is filled (for example an attribute for a UF, or name field for a NF). When the request is executed against a field which belongs to the first entry of an array, the specified parameters are propagated to the other entries of the array.
- The ENTER ARRAY request duplicates a given number of lines, starting at a specified line, a certain number of times in order to generate an array. You must specify the array name and whether the whole lines must be duplicated or only their named fields. This last option is allowed only if both the first and last line of the array contain a named field.



2.2 MODIFY Command

Purpose:

This command allows you to dynamically update the definition of a form in full screen mode. Although this command does not involve the use of form definition statements, it can access most options available to the form definition language, as described below. The options that may not be specified during the process of the MODIFY command will be set as in the original form.

Syntax:

```
MODIFY form-name1 [NEW=form-name2]

        { terminal-type [,terminal-type ] ... }
[TERM = { ANY                } ]
        { ALL                }

[LANG={COBOL | CL | GPL | DD}] [FORCE]

[RECORD=record-name];
```

where terminal-type =

```
{DKU7007|DKU7005|IBM3270|VIP7804|DKU7107|DKU7211|MINITEL|
PC7800|IBM3278|IBM3279}
```



Parameters:

form-name1 the name of the form to be updated.

NEW The name of the form to be generated as a result of the update. If not specified, it is assumed to be form-name1.

TERM The terminal type or list of terminal types (separated by commas) for which the object form must be updated. When ANY is specified, instead of generating a terminal-dependent object member, MNFORM generates a terminal-independent object member which is adapted to the terminal at run-time.

This terminal-independent object member cannot be moved to the UFAS outfile or to the diskette.

When ALL is specified, the form is updated for all terminal types that are supported.

If no terminal type is specified, a default one is given. In BATCH, the default is VIP7700. In IOF, the default is the type of the terminal on which the user is logged, or VIP7700 if the terminal does not support FORMS run time package.

MNFORM stores in the form-identification structures a flag that indicates whether or not the ANY option has been selected. When the form is accessed at run time, and if ANY was not selected, FORMS looks in each binary library, first for the terminal-dependent member, then if not found for the terminal-independent member. If ANY was selected, FORMS looks first for the terminal-independent member, then for the terminal-dependent member.

The terminal type IBM3278 is mapped in IBM3279. When several terminal-types exist for a given form, it is possible to modify this form for a single terminal-type. But this operation may lead to an incoherence among the terminal-types members which are formname-of-terminal-type.

For instance, if form-name-of-DKU7007 and form-name-of-IBM3270 both exist, and if form-name-of-DKU7007 is modified, the members form-name-of-DKU7007 and form-name-of-IBM3270 become different.



We recommend that the form be modified for all existing terminal-types.

This recommendation is also valid when ANY-terminal-type and others terminal-types simultaneously exist for a given form.

LANG	<p>(alias TYPE). The language for which language members are to be updated.</p> <p>When DD is specified, a data description member is generated in BINLIB. No language members are generated in SLLIB. DD should be used only for forms to be used with IQS.</p>
RECORD	<p>This is meaningful when LANG=DD. It indicates the name of the record to be retrieved by IQS. When RECORD is not specified, the form name is taken as the record name. The name of the data description member is record-name D.</p>
FORCE	<p>means that the program referencing the form will have to be recompiled.</p> <p>There are three kinds of operations on named fields: modification of parameters for already existing fields, deletion of old fields and addition of new fields. It must be noted that the last two operations will imply a recompile of the programs referencing the form. Therefore, they are allowed only when the FORCE option is specified. Modification of any of the following named field parameters also necessitates a re-compile: name, length, or screen-picture. On the other hand, since unnamed fields are not known to application programs, they are not subject to this limitation and may be changed without restriction. You must use the FORCE option when you make any of the following changes: add or delete an array; change the dimensions of an array; change the list of named fields in an array; or modify the array generation for DKU terminals by changing the value of ALLOW RPLF. When FORCE is not selected, the order of the fields in the data record is not changed, even in case of field permutation on the screen.</p>



2.2.1 Command Processing

The MODIFY process consists of the following phases:

- specification phase,
- delete phase,
- drawing phase,
- description phase.

2.2.1.1 Specification Phase

In the specification phase, you can specify the same general purpose parameters as in the specification phase of the ICREATE command. The default characters are those used in the old form. If the FORCE option has been specified, you should also specify whether any named fields are to be deleted.

2.2.1.2 Delete Phase

The delete phase takes place only if you have specified that named fields are to be deleted. An image of the form is displayed on the screen with the conventions on NF and UF specified in the first phase. You must erase with spaces the fields to be deleted without performing any other modification at this time and retransmit the image. You are then notified of the number of deleted fields, for checking purposes.

2.2.1.3 Drawing Phase

In the drawing phase, an image of the form is displayed on the screen, without the deleted fields and with the conventional characters specified in the first phase. The unnamed field starting and ending characters are displayed for the unnamed fields with a rendition attribute. You may perform any modification you want, such as changing the unnamed fields, adding new named fields (only if FORCE option) or displacing existing named fields. Then you should transmit again the image. At this time, the attributes relative to the unnamed fields of the original form are automatically applied to the corresponding unnamed fields of the new form (First, the UFs at the same location with the same value are mapped, then the remaining UFs are mapped going from top to bottom, right to left).



2.2.1.4 Description Phase

The purpose of the description phase is to map the parameters of the NF as defined in the original form onto the new image of the form and to specify for which fields some parameters have to be changed.

In this phase, the screen is divided into two parts, in the same way as in the corresponding phase of the ICREATE command. The upper part of the screen displays a window on the form drawing whereas the lower part contains one of the three definition forms (NF, UF and ARRAY). In this phase, you are supposed to issue an ENTER NF request for each named field that is either new or whose relative position in the screen has changed. The mapping of the non deleted fields of the original form onto the fields of the new form whose position has not changed may then be automatically performed through the MAP request. The following requests are available:

- **ENTER NF,UF, or ARRAY:** When the form of the lower part of the screen does not match the type of item to be entered, the result of this request is to replace this form by the appropriate one. Otherwise, this request enters the parameters specified in the form in the description of the referenced item.

An NF or UF is referenced by its line number and its rank within the line. When ENTER NF applies to a field that existed in the original form, (which is the only case allowed if FORCE was not specified), the "name" field of the NF form must contain the name of the field. This field may have been filled by a previous RETRIEVE request. If the "name" field is not filled or contains a name that is not the name of a field in the original form, it is assumed that a new field is to be entered.

The ENTER ARRAY request allows you to perform the same operations at array level as the ENTER NF at NF level. If the array name contains the name of an array in the original form, the request maps this old array. Otherwise, it assumes that a new array is to be entered. This last case is allowed only in the case of the FORCE option.

- **RETRIEVE:** This request retrieves information relative to a specified field (or array). A UF is referenced by (line, rank), an NF either by name or by (line, rank), an array by name or starting line.

In the case of a retrieve by name, the field (or array) is first looked for in the new description of the form. If found, the corresponding parameters are displayed, including the location of the item (line and rank). Otherwise, the item is searched in the description of the original form. In this case, the RETRIEVE request sets to zero the location parameters (line, rank). A retrieve by name request is recommended before modifying the length or the position of an NF so as to avoid losing the other characteristics of the field.



- **MAP:** This request maps information relative to the fields of the original form onto the corresponding fields of the new form. If it has not been explicitly invoked, it is automatically performed at the end of the MODIFY process. In other words, the named fields of the original form that have not been deleted nor re-entered in the new form are scanned, in screen order (from left to right, top to bottom). At the same time, the fields of the new form that were not subject to an ENTER request are scanned and the field parameters of the old form are applied to the corresponding fields (same location and same length) of the new form. When an array is encountered, it is expanded.
- **TEST:** This request mounts the form being generated on the screen. Depressing the TRANSMIT key will restore the screen to its previous contents.
- **QUIT:** This request completes the command. You are asked whether you want to save the updated form.

The preceding requests are expected to be used under the following conditions:

- No request is needed against deleted fields, since the delete operations are handled in a specific phase.
- In the case of the FORCE option, if a new NF has been added in the drawing phase, an ENTER NF must be issued against this field. The NF name need not be specified but, if specified, it must be different from the names of the fields in the original form.
- For each NF which has been displaced and/or the length of which has been changed, one should issue a Retrieve By Name request in order to get the field parameters, modify parameters and issue an ENTER NF in order to indicate the new location of the field.
- For each array which has been displaced relatively to the other NFS, one should issue a RETRIEVE ARRAY by name in order to get the array parameters and an ENTER AR in order to indicate the new location of the array. If a field belonging to an array has to be modified, an ENTER AR must be issued before issuing the corresponding ENTER NF.
- After these actions are completed, the MAP request should be issued in order to map the parameters of the NFs of the original form onto the corresponding NFs of the new form and to expand the arrays not yet expanded.
- Then for each NF which has not been displaced but for which some parameter has to be changed, one should issue a Retrieve (by location or by name), modify the parameters and issue an ENTER NF.
- An ENTER UF may be issued at any time against a UF for which the implicit attribute mapping is not satisfactory. (This mapping may be checked at any time through the TEST request).



The order in which the requests are invoked in the scenario described above is indicative except that the MAP request must not be invoked before an ENTER NF has been issued for all new or displaced fields and an ENTER AR has been issued for all new or displaced arrays. Moreover, a Retrieve by location on a field for which no ENTER NF has been issued is possible only after the MAP request.

2.2.1.5 Methods of Modifying Forms

The following scenario details the order in which requests must be invoked, depending on whether or not the FORCE option has been selected. Be warned that issuing a MAP request before specifying all new or displayed fields and arrays may lead to problems.

The FORCE option is not used:

The following modifications are possible:

unnamed fields added or deleted
attributes changed.

1. Specification phase
2. Drawing phase:
 - fields and arrays may be displaced.
 - unnamed fields may be added or displaced.
3. Before using the MAP request, you must specify the new location of the fields or arrays which have been displaced and/or the lengths of which have been changed. You must retrieve the fields and arrays by name and re-enter them with their new location or length (ENTER NF ENTER ARRAY).
4. You may then use the MAP request to map information of the fields of the original form onto the corresponding fields of the new form.
5. Named or unnamed field attributes may be changed using the RETRIEVE request (by location or name) followed by an ENTER NF or ENTER UF.



The FORCE option is used:

All modifications are allowed.

1. Specification phase
2. Delete initial phase
3. Drawing phase
4. Retrieve by name displaced fields or arrays. Enter the new location (ENTER NF, ENTER ARRAY), as if FORCE is not used.
5. Definition of new fields or arrays.
 - ENTER NF (with no name or a new name).
 - ENTER ARRAY (a new name is mandatory).
6. Mapping of information in fields of the original form onto the corresponding fields of the new form (MAP request).
7. Named or unnamed field attributes may be changed using the RETRIEVE request (by location or name) followed by an ENTER NF or ENTER UF.

2.2.1.6 Warning

When modifying a form containing two consecutive protected Named Fields, FORMS cannot recognize a separation between these 2 NF, due to the drawing presentation screen, and will make a concatenation. Then:

- If the FORCE option is not used, MODIFY will be stopped after drawing phase and the specification screen will be displayed with the message " * LENGTH OF FIELD CHANGED, USE FORCE OPTION ".
- If the FORCE option is used, the two NF will be changed into a single NF.



2.3 CREATE Command

Purpose:

This command is used to create a form using form definition statements as defined above.

In IOF, the terminal is turned to request mode, so that you may enter the form definition statements.

In Batch these statements must follow the CREATE statement.

The end of the CREATE command is obtained by the END CREATE (or END) statement if you want to save the form, or by QUIT or / if not. In Batch, the END CREATE statement must follow the form definition statements.

Members generated:

source member (BINLIB)
terminal object members (BINLIB)
language members (SLLIB)

Syntax:

CREATE form-name

```
[TERM = { terminal-type [,terminal-type ] ... }
        { ANY } ]
        { ALL } ]
```

[LANG = {COBOL | CL | GPL | DD}]

[RECORD=record-name];

where terminal-type =

```
{DKU7007 | DKU7005 | IBM3270 | VIP7804 | DKU7107 | DKU7211 | MINITEL |
PC7800 | IBM3278 | IBM3279}
```



Parameters:

form-name The name of the form to be created. Its length must be less than or equal to 8 characters. The first one must be alphabetic and the others alphanumeric. The minus character is allowed too.

TERM The terminal type or list of terminal types (separated by commas) for which the object form must be loaded. When ALL is specified, the form is loaded for all terminal types that are supported. When ANY is specified, instead of generating a terminal-dependent object-member, MNFORM generates a terminal-independent object member which is adapted to the terminal at run-time. This terminal-independent object member cannot be moved to the UFAS outfile or to the diskette. When no terminal type is specified, a default one is given. In batch, the default is VIP7700. In IOF, the default is the type of the terminal on which the user is logged or ANY if the terminal does not support the FORMS run time package.

MNFORM stores in the form-identification structures a flag that indicates whether or not the ANY option has been selected. When the form is accessed at run-time, and if VIP7700 was not selected, FORMS looks in each binary library, first for the terminal-dependant member, then if one is not found for the terminal-independent member. If ANY was selected, FORMS looks first for the terminal-independent member, then for the terminal-dependent member.

The terminal type IBM3278 is mapped in IBM3279.

When several terminal-types exist for a given form, it is possible to modify this form for a single terminal-type. But this operation may lead to an incoherence among the terminal-types members which are formname-of-terminal-type.

For instance, if form-name-of-DKU7007 and form-name-of-IBM3270 both exist, and if form-name-of-DKU7007 is modified, the members form-name-of-DKU7007 and form-name-of-IBM3270 become different.



We recommend that the form be modified for all existing terminal-types.

This recommendation is also valid when VIP7700-terminal-type and other terminal-types simultaneously exist for a given form.

LANG

(alias TYPE) the language for which language members are to be loaded. The default is the value of the #FORMLANG system variable.

When DD is specified, a data description member is generated in BINLIB. No language members are generated in SLLIB. DD should be used only for forms to be used with IQS.

RECORD

This is meaningful when LANG=DD. It indicates the name of the record to be retrieved by IQS. When RECORD is not specified, the form name is taken as the record name. The name of the data description member is record-name D.



2.4 EDIT Command

Purpose:

This command uses the Text Editor on SLLIB. For more information, refer to the *Text Editor User's Guide*.

Syntax:

EDIT;

Parameters:

None

2.5 FSE Command

Purpose:

This command uses the Full Screen Editor on SLLIB. For more information, refer to the *Full Screen Editor User's Guide*.

Syntax:

FSE;

Parameters:

None



2.6 COMPILE Command

Purpose:

This command is used to generate a form from a source member in SLLIB created or updated with the EDIT or FSE commands.

The source member must have the same name as the form to be generated and contain source definition language statements, as defined above.

If the terminalist does not want the source form definition language statement to be displayed on the terminal during form compilation, the BRIEF option must be specified.

The BRIEF option has no effect in batch mode. It is ignored when set.

If the form already exists, either the REPLACE or the FORCE option must be specified.

Members generated:

source member (BINLIB)
terminal object members (BINLIB)
language members (SLLIB).

Syntax:

COMPILE form-name

```
[TERM = { terminal-type [,terminal-type ] ... }
        { ANY } ]
        { ALL } ]
```

```
[LANG = {COBOL | CL | GPL | DD}] [REPLACE FORCE] [BRIEF]
```

```
[RECORD=record-name];
```

where terminal-type =

```
{DKU7007|DKU7005|IBM3270|VIP7804|DKU7107|DKU7211|MINITEL|
PC7800|IBM3278|IBM3279}
```




Parameters:

form-name

The name of a member in SLLIB that contains form definition statements to generate a form with the same name. Its length must be less than or equal to 8 characters. The first one must be alphabetic and the others alphanumeric. The minus character is allowed too.

TERM

The terminal type or list of terminal types (separated by commas) for which the object form must be loaded. When ANY is specified, instead of generating a terminal-dependent member, MNFORM generates a terminal-independent member which is adapted to the terminal at run-time. This terminal-independent object member cannot be moved to the UFAS outfile or to the diskette. When ALL is specified, the form is loaded for all terminal types that are supported. When no terminal type is specified, a default one is given. In batch, the default is VIP7700. In IOF, the default is the type of the terminal on which the user is logged or ANY if the terminal does not support FORMS run time package.

MNFORM stores in the form-identification structures a flag that indicates whether or not the ANY option has been selected. When the form is accessed at run-time, and if ANY was not selected, FORMS looks in each binary library, first for the terminal-dependent member, then if one is not found for the terminal-independent member. If ANY was selected, FORMS looks first for the terminal-independent member, then for the terminal-dependent member.

The terminal type IBM3278 is mapped in IBM3279.

When several terminal-types exist for a given form, it is possible to modify this form for a single terminal-type. But this operation may lead to an incoherence among the terminal-types members which are form-name-of-terminal-type.

For instance, if form-name-of-DKU7007 and form-name-of-IBM3270 both exist, and if form-name-of-DKU7007 is modified, the members form-name-of-DKU7007 and form-name-of-IBM3270 become different.



We recommend that the form be modified for all existing terminal-types.

This recommendation is also valid when ANY-terminal-type and other terminal-types simultaneously exist for a given form.

LANG

(alias TYPE) the language for which language members are to be loaded. The default is the value of the #FORMLANG system variable.

When DD is specified, a data description member is generated in BINLIB. No language members are generated in SLLIB. DD should be used only for forms to be used with IQS.

RECORD

This is meaningful when LANG=DD. It indicates the name of the record to be retrieved by IQS. When RECORD is not specified, the form name is taken as the record name. The name of the data description member is record-name D.

REPLACE

means that already existing members may be replaced by new ones, provided that no recompilation of the programs referencing the form is necessary.

A recompilation will be necessary if one of the following points has changed:

- Number of named fields in the form.
- Array dimension.
- Array name.
- Named field name.
- Named field length.
- Named field screen picture.

If a named field has no name defined, default is the name existing before the update.

If no picture is defined, default is the picture existing before the update.

FORCE

means that already existing members will be replaced by new ones and that the program referencing the form will have to be recompiled.



BRIEF

indicates whether or not the source form definition language statements are to be displayed on the terminal during form compilation.

IF BRIEF = 0 (default option) is specified, the source form definition language statements are to be displayed on the terminal.

IF BRIEF = 1 is specified, the source form definition language statements are not to be displayed on the terminal. Only the erroneous statements, followed by their appropriate error messages are displayed. If an erroneous statement is split into several lines, only the last line of this statement is displayed before the error message.



2.7 DECOMPILE Command

Purpose:

This command builds a form source member in SLLIB from the reference source member in BINLIB. This member has the same name as the form it defines. It may be modified by FSE and EDIT commands in order to be resubmitted through COMPILE command.

Syntax:

```
DECOMPILE form-name [REPLACE];
```

Parameters:

form-name	The name of the form to be decompiled. A star convention is accepted.
REPLACE	means that an already existing source in SLLIB will be replaced by the decompiled one.



An extracted form is considered as a form without a named field.

The associated data record contains only the implicit function key field. To use this form to input data other than a function key at run time, you must overlay it with a form created by `MAINTAIN_FORM` which contains named fields. For more details, see the `OVERLAY` mode in the `CDGET` primitive.

Syntax:

`EXTRACT form-name`

`FROM=member-name, TERM=MINITEL`

`[LANG={COBOL | CL | GPL | DD}]`

`[RECORD=record-name]`

`[REPLACE];`

Parameters:

form-name	The name of the form to be extracted. The name must be less than or equal to 8 characters. The first one must be alphabetic and the others alphanumeric. The minus character is allowed too. The star convention is accepted.
TERM	The terminal type for which the object form must be loaded. Only the value <code>MINITEL</code> is accepted.
LANG	(alias <code>TYPE</code>) the language for which language members are to be loaded. The default is <code>COBOL</code> . When <code>DD</code> is specified, a data description member is generated in <code>BINLIB</code> . No language members are generated in <code>SLLIB</code> . <code>DD</code> should be used only for forms to be used with <code>IQS</code> .
RECORD	This is meaningful when <code>LANG=DD</code> . It indicates the name of the record to be retrieved by <code>IQS</code> . When <code>RECORD</code> is not specified, the form name is taken as the record name. The name of the data description member is <code>record_name_D</code> .
REPLACE	means that already existing objects will be replaced by new ones.



3. Managing Forms

3.1 DELETE Command

Purpose:

This command deletes the form members or records in the file or library specified. Note that, for this command, BINLIB stands for BINLIB and SLLIB.

Syntax:

```
DELETE form-name  
  
    [FROM ={BINLIB OUTFILE DK} [DSN =data-set-name]]  
  
    [FORCE];
```

Parameters:

form-name	The name of the form to be deleted. A star convention is accepted if the FROM parameter specifies BINLIB.
FROM	The file or library from which the form must be removed. If this parameter is missing, BINLIB is assumed.
BINLIB	All the members in BINLIB and in SLLIB are deleted, except the form source members, which are deleted only when FORCE is specified. The data description members generated for IQS (when LANG=DD is specified at creation time) are not deleted by the DELETE command.



OUTFILE	All records relative to the form are deleted from the UFAS output file.
DK	The form is deleted from the diskette, if applicable. If DSN is not specified, the data set name is retrieved from the object form member in BINLIB.
DSN	Data set name (refer to DK). It length must be less than 8 characters. The first one must be alphabetic and the others alphanumeric. The minus character is allowed too.
FORCE	If FROM=BINLIB, it means that all members relative to the form must be deleted in BINLIB and SLLIB, including the source form members. If FROM=DK, it means that the object form in BINLIB must be flagged as "not moved to diskette"; i.e., that the FORMS run time package will not assume that this form is subject to local storage.



3.2 DISPLAY Command

Purpose:

This command displays an image of the form built from the reference source member in BINLIB.

In this image, the named fields are replaced by the named field character used in the last pictorial description of the form, or by # if not applicable.

Syntax:

```
DISPLAY form-name ;
```

Parameters:

form-name	The name of the form to be displayed. A star convention is accepted.
-----------	----------------------------------------------------------------------



3.3 LIST Command

Purpose:

This command produces a list of the objects relative to the specified form in the file or library selected. It enables to check for which terminal types a form has been loaded. When ALL is specified, it gives the list of all the forms contained in the file or library.

Syntax:

```
LIST {form-name ALL}
      [FROM = {BINLIB [NAMES] SLLIB OUTFILE DK} [DSN=data-set-name]];
```

Parameters:

form-name	The name of the form for which members are to be listed. A star convention is accepted if the FROM parameter specifies BINLIB or SLLIB. If ALL is used, all the forms in the specified file or library are listed, possibly in a shortened way.
FROM	The file or library from which members are to be listed. If not specified, BINLIB is assumed.
BINLIB	The object and source members of BINLIB are listed. If NAMES is specified, the listing is restricted to the names of the members. The data description members generated for IQS are not listed.
SLLIB	All members in SLLIB whose name start with the name of the form are listed.
OUTFILE	The fixed form (.F) and data map (.M) records relative to the form in OUTFILE are listed.



- DK This option is supported only under IOF from a terminal with a diskette.
- If ALL is required and no DSN is specified, the DSN catalog is listed.
 - If ALL is required and a DSN is specified, the forms in this data set are listed.
 - If a form name and a DSN are specified, this form is listed within this data set.
 - If a form name is specified and no DSN is specified, the data set name is retrieved from the object form member. If not found, DS1 is assumed.
- DSN Data set name (refer to DK). Its length must be less than 8 characters. The first one must be alphabetic and the others alphanumeric. The minus character is allowed too.



3.4 LOAD Command

Purpose:

This command generates the appropriate terminal object members and language members for an already created form. It enables a form to be used for other terminal types and languages than the ones specified at form creation time. It can also be used in case of loss of these members to rebuild them, as long as the reference source member in BINLIB is present.

Syntax:

```
LOAD form-name

    [TERM ={terminal-type ANY ALL (terminal-type,...)}]

    [LANG ={COBOL | CL | GPL | DD}] [{REPLACE FORCE}]

    [RECORD=record-name];
```

where terminal-type =

```
{DKU7007 | DKU7005 | IBM3270 | VIP7804 | DKU7107 | DKU7211 | MINITEL
    | PC7800 | IBM3278 | IBM3279}
```

Parameters:

form-name	The name of the form to be loaded. A star convention is accepted.
TERM	The terminal type or list of terminal types (separated by commas) for which the object form must be loaded. When ANY is specified, instead of generating a terminal-dependent member, MNFORM generates a terminal-independent member which is adapted to the terminal at run-time. This terminal-independent object member cannot be moved to the UFAS outfile or to the diskette. When ALL is specified, the form is loaded for all terminal types that are supported. When no terminal type is specified, a default one is given. In batch, the default is VIP7700. In IOF, the default is the type of the terminal on which the user is logged or ANY if the terminal does not support FORMS run time package.



MNFORM stores in the form-identification structures a flag that indicates whether or not the ANY option has been selected. When the form is accessed at run-time, and if ANY was not selected, FORMS looks in each binary library, first for the terminal-dependent member, then if one is not found for the terminal-independent member. If ANY was selected, FORMS looks first for the terminal-independent member, then for the terminal-dependent member.

The terminal type IBM3278 is mapped in IBM3279.

When several terminal-types exist for a given form, it is possible to modify this form for a single terminal-type. But this operation may lead to an incoherence among the terminal-types members which are form-name-of-terminal-type.

For instance, if form-name-of-DKU7007 and form-name-of-IBM3270 both exist, and if form-name-of-DKU7007 is modified, the members form-name-of-DKU7007 and form-name-of-IBM3270 become different.

We recommend that the form be modified for all existing terminal-types.

This recommendation is also valid when ANY-terminal-type and other terminal-types simultaneously exist for a given form.

LANG

(alias TYPE) the language for which language members are to be loaded. The default is the value of the #FORMLANG system variable.

When DD is specified, a data description member is generated in BINLIB. No language members are generated in SLLIB. DD should be used only for forms to be used with IQS.



RECORD	This is meaningful when LANG=DD. It indicates the name of the record to be retrieved by IQS. When RECORD is not specified, the form name is taken as the record name. The name of the data description member is record-name D.
REPLACE or FORCE	means that existing terminal object members or language members will be replaced by new ones. If REPLACE or FORCE is omitted, existing terminal object members or language members are not modified. The flag Formname-AF is not modified either.



3.5 MOVE Command

Purpose:

This command is used to move the terminal object form members from BINLIB to OUTFILE or to the diskette. This latter option is only available from an IOF terminal with a diskette (only DKU7007, DKU7107 and VIP7760).

The terminal-independent object member (Form-name-OF-ANY) cannot be moved to the UFAS outfile or to the terminal diskette.

For certain terminals, when a MOVE command is executed to a diskette, the data set name is stored with the object form member in BINLIB. At run-time, this data set name is retrieved and whenever an attempt is made to mount a form with the appropriate terminal type and with a diskette, a control sequence requesting the local mounting of the form is issued. When such a facility is used, the form must be supplied to all terminals with a diskette in the network, either by a series of MOVE commands or by duplicating the diskette and distributing it.

An additional option is provided to build object form members in BINLIB from the form records of OUTFILE.

Note that at move time, any PLE or PLW characters that do not belong to the C101 character set are converted into the closest basic characters.

Syntax:

```
MOVE {form-name ALL}

[FROM ={BINLIB [TO={DK[DSN=data-set-name] OUTFILE}] OUTFILE}]

[REPLACE];

(alias STORE)
```

**Parameters:**

form-name	The name of the form for which members are to be moved. A star convention is accepted if the FROM parameter specifies BINLIB. If ALL is used, all the forms in the specified file or library are moved (This option is not available to move forms from BINLIB to DK).
FROM	The file or library from which members are to be moved. If not specified, BINLIB is assumed.
BINLIB	The object members of BINLIB are moved according to the TO parameter.
OUTFILE	Terminal object members of the form are built in BINLIB from the fixed form (.F) and data map (.M) records existing in OUTFILE.
TO	The file to which the form is moved (only applicable if FROM= BINLIB). If not specified, DK is assumed.
DK	This option is supported only under IOF from a terminal with a diskette. This option does not support the ALL keyword. The object form for the terminal under which the user is connected is moved to diskette, when applicable. The maximum size of a binary object form moved to a diskette is 4Kbytes. For biggest form, the move command is rejected with the error message: "TOO LONG OBJECT PICTURE. IGNORED".
OUTFILE	The fixed form (.F) and data map (.M) records relative to the form are built from terminal object members.
DSN	The data set name under which the form has to be moved. When not specified, if the form is already flagged as "moved to diskette", the DSN is retrieved from the object form member in BINLIB; else DS1 is assumed. The length of data set name must be less than 8 characters. The first one must be alphabetic and the others alphanumeric (The minus character is allowed too).
REPLACE	means that already existing objects will be replaced by new ones.



3.6 PRINT Command

Purpose:

This command prints the reference source member from BINLIB.

Syntax:

```
PRINT form-name;
```

Parameters:

form-name	The name of the form to be printed. A star convention is accepted.
-----------	--------------------------------------------------------------------

3.7 QUIT Command

Purpose:

This command causes MAINTAIN_FORM to exit. After the exit, the IOF terminal is at system level.

Syntax:

```
QUIT;
```

Parameters:

None



3.8 TEST Command

Purpose:

This command mounts the form on the screen, as it will appear at run time. Then you can enter data in the variable fields and transmit the screen. The contents of the resulting data record are displayed on the screen.

Syntax:

```
TEST form-name;
```

Parameters:

form-name The name of the form to be tested.

3.9 TRANSIT Command

Purpose:

This command is a transit tool for forms created under GCOS64 release 1E. It builds a reference source member in BINLIB from the 1E source member in SLLIB. A message is issued for those which are not recognized as 1E source members.

Syntax:

```
TRANSIT {form-name|ALL} [REPLACE];
```

Parameters:

form-name The name of the form to be transferred. When ALL is specified, all members whose name length does not exceed 8 characters are attempted to be transferred.

REPLACE Implies that the reference source has to be generated, even if it already exists.



4. Source Form Definition Language

This section describes the source form language that is to be used to define a form through a CREATE or COMPILE command and that is generated by MAINTAIN_FORM in the reference source form member. Multiple statements may be specified in a record, and statements may be continued across records. A keyword or literal may not be continued across a record.

The statements allowed in a source form are shown below.

4.1 General Format

```
[NO IMPLICIT {FUNCTION-CODE|FC}.]  
  
[  
    { SPACES } ]  
[RECEIVE { }].]  
[  
    { SPACE } ]  
  
[RECEIVE REAL-MODE.]  
  
[ALLOWRPLF.]  
  
[SUBSTITUTE ATTRIBUTES.]  
  
[CURRENCY SIGN IS literal.]  
  
[DECIMAL-POINT IS COMMA.]  
  
[DESC literal.]  
  
[NFC literal.]  
  
[form pictorial description]  
  
[ { UFIELD / UF } ]  
[ { } or ARRAY description entry 1 . . . ]  
[ { NFIELD / NF } ]
```

*comments



4.1.1 Syntax Rules

1. The statements in the form description must appear in the order specified above.
2. Form-name is a character string formed from the set of characters A through Z and 0 through 9. The first character of the form-name must be an alphabetic character. The form-name is used as the basis of the language member names and the object form member names. If the form-name exceeds eight characters, only the first eight characters are used.
3. Each statement must be terminated by a period or a semi-colon.
4. The equal sign (=) is accepted as an alias of IS in the statements.

4.1.2 General Rules

1. A set of comment lines may be specified after each UF or NF statement. Comment lines are for documentation purposes only and are not compiled by MAINTAIN FORM. A comment line begins with an asterisk and may contain any character.
2. The NO IMPLICIT FUNCTION-CODE option means that the explicit function code field is not generated at the head of the data record and selection vector structures. When this option is selected, the function codes are not received except if a NF with a FUNCTION clause is explicitly declared.
3. The RECEIVE SPACES option means that the fields are declared as "received" in the data record when they are transmitted even if they are empty or filled with spaces. In such cases, the data record field contents is set to spaces for a field with an alphanumeric PICTURE clause and to zero for a field with a numeric PICTURE clause. This option and RECEIVE REAL-MODE are mutually exclusive.
4. The RECEIVE REAL-MODE option means that the fields are declared as "received" in the data record when they are not empty and are actually transmitted by the terminal. This clause is provided for compatibility with previous releases; its use is not recommended since it breaks terminal independence. When this clause is not specified, a field is declared as "received" when it is transmittable, not empty, and its contents are different from spaces.
5. The ALLOW RPLF option means that arrays for the DKU terminal range must be generated with RPLF terminal functionality. When this functionality is used, only the first line of an array is generated in the fixed form. The other lines are automatically copied by the terminal. Such forms are activated only in Append mode at the top of the screen.



6. The SUBSTITUTE ATTRIBUTE option indicates that, when a rendition attribute is specified on some field and this attribute is not supported by a terminal for which the form is to be loaded, this attribute may be replaced by another attribute. Otherwise, the attribute is ignored. The substitutions actually performed are detailed under "Terminal Dependent Features" in this section.
7. The CURRENCY SIGN IS literal option substitutes the specified literal for the dollar sign in the character-strings of SCREEN-PICTURE clause and in numeric literals. (Note for COBOL programs, this optional phrase must also be specified in the ENVIRONMENT DIVISION).
8. The DECIMAL-POINT IS COMMA statement means the functions of the comma and period are interchanged in the character strings of the SCREEN-PICTURE clause and in numeric literals. (Note For COBOL programs, the DECIMAL-POINT IS COMMA phrase must also be specified in the ENVIRONMENT DIVISION.)
9. The DESC statement is used to enter a text that will be included as a comment in one of the language members. If the text is more than 60 characters long, it will be truncated.
10. NFC is an abbreviation for NAMED-FIELD-CHARACTER. This statement specifies the named-field character, the end of named-field character, and the starting and ending unnamed-field characters. One to four characters can be supplied. If more than four characters are given in this entry, then only the first four are taken into account.

Character-1 is used in the form pictorial description to represent the positions of the named fields.

Character-2 is used to signal the end of a named field, for VIP 7700/7001/7002 terminals. If it is not given, the default character is a period (.).

Character-3 and character-4 are used respectively to specify the starting and the ending unnamed-field-characters for the form pictorial description. Semi-colon (;) and period (.) may not be used for character-3 and character-4.

If this request is not used, the default characters are:

; < >



4.2 Pictorial Description

The pictorial description of a form consists of a set of pictorial description entries which define named and unnamed fields. The pictorial description must be preceded by the START-FORM statement and terminated by the END-FORM (or [F]) statement.

4.2.1 General Format

```
START-FORM.
```

```
{pictorial-description-entry}...
```

```
{END-FORM / [F]}
```

4.2.2 Syntax Rules

1. The whole pictorial description of the form is optional; the form definition may be completely specified by the NFIELD/UFIELD statements.
2. If the START-FORM statement is specified, the pictorial description entries and the END-FORM (or [F]) statement are required.

4.2.3 General Rules

1. Each pictorial description entry provides the image of a form line exactly as it should appear on the terminal device.
2. Each named field in a pictorial description statement is defined by a contiguous sequence of character-1 (as given in the NFC statement) or # if the NFC statement is omitted.
3. Each unnamed field in a pictorial description entry is defined by a contiguous character string formed from any character in the computer character set except character-1 of the NFC statement (or # if the NFC statement is omitted).



4.3 UFIELD/NFIELD Statements

4.3.1 General Format

{UFIELD/UF} location value [length] [attributes].

{NFIELD/NF} [data name] [location] [length] [picture]

[usage][attributes].

{NFIELD/NF} [data name] {FUNCTION-CODE/FC}.

4.3.2 General Rules

1. The data name clause, if specified, must be the first clause in an NFIELD statement. The order of the other clauses is not significant.
2. NFIELD and UFIELD statements may not define overlapping fields or fields spreading over a line boundary.
3. The size of a UFIELD is determined by either:
 - a) The integer in the LENGTH clause, if specified.
 - b) The length of the literal in the VALUE clause, if the LENGTH clause is not specified.
4. The size of a NFIELD is determined by either the integer in the LENGTH clause if specified or the SCREEN-PICTURE clause, if specified.
5. The order of the fields in the generated data record and selection vector structure is the order of the NF statements.
6. The maximum number of named fields is 254 if the NO IMPLICIT FUNCTION CODE clause is not specified, 255 otherwise. This number includes the field repetitions in an array. Some additional limitations may apply, depending on terminal types. These limitations are detailed under "Terminal Dependant Features" below.
7. It is recommended to leave a separating space (1 column) between two consecutive Named Fields. Otherwise, the ICREATE and MODIFY interactive commands cannot recognize a separation and will concatenate the two NF into a single NF (see section 2.2.1.6).



4.3.3 Data-name Clause

1. Data-name is a character string of 1 to 30 characters, formed from the set of characters A through Z, 0 through 9, and the hyphen. The first character must be alphabetic.
2. Data-name is the name given to this field for the data record and the root of the name for the selection vector.
3. The data-name clause is optional. When omitted, the standard name form-name-nnn will be used, nnn standing for the rank of the named field in the data record nnn is 01 through 255 (e.g. the name of the first field is form-name-01.)
4. All data-names within a form must be unique.

4.3.4 Location Clause

Format 1:

LINE IS integer-1 [{COLUMN/COL} IS integer-2]

1. The LINE clause must be specified for either NFIELD or UFIELD statements when there is no form pictorial description.
2. The LINE clause is used to specify the position of the field relative to the beginning of the form. The upper left hand corner of the form is LINE IS 1 COLUMN IS 1.
3. The COLUMN phrase of the LINE clause is optional and if omitted, the value of the COLUMN is 1.
4. Integer-1 cannot exceed 24 and integer-2 cannot exceed 80.

Format 2:

FLD IS integer-3

1. The FLD clause is restricted to NFIELD statements when there is a form pictorial description.
2. Integer-3 refers to the rank of the NFIELD in the form pictorial description starting from the left-hand corner, left to right and top to bottom.



4.3.5 LENGTH Clause

{ LENGTH | LEN } IS integer

1. The LENGTH clause must not contradict the picture if specified and if the LENGTH clause and the VALUE clause are both specified, the integer in the LENGTH clause must be greater than or equal to the length of the literal specified in the VALUE clause.
2. Integer may not exceed 80.

4.3.6 SCREEN-PICTURE Clause

The SCREEN-PICTURE clause describes the view of the data in the screen.

General Format

{ SCREEN-PICTURE }
{ SCREEN-PIC } IS character-string

Syntax Rules

1. The maximum number of characters in the character-string is 18.
2. The SCREEN-PICTURE clause defines the format of the data in the screen.

General Rules

1. There are three categories of data that can be described with a SCREEN-PICTURE clause: numeric, alphanumeric, and numeric edited.
2. To define an item as numeric:
Its SCREEN-PICTURE character-string is restricted to the symbol '9'.
3. To define an item as alphanumeric:
Its SCREEN-PICTURE character-string is restricted to the symbol 'X'.



4. To define an item as numeric edited:
 - Its SCREEN-PICTURE character-string is restricted to certain combinations of the symbols 'z', '*', '9', ',', '.', '+', '-' and the currency symbol. The allowable combinations are determined from the order of precedence of symbols and the editing rules.
 - The character-string must contain at least one 'Z', '*', '+', '-', ',', '.' or the currency symbol.
5. The size of an elementary item, where size means the number of character positions occupied by the item in Standard Data Format, is determined by the number of allowable symbols that represent character positions. An integer which is enclosed in parentheses following the symbols 'X', 'Z', '*', or '9', indicates the number of consecutive occurrences of the symbol. Note that the following symbols may appear only once in a given SCREEN-PICTURE: '.', '+', '-' or 'cs'.
6. The functions of the symbols used to describe screen items are explained as follows:

X Each 'X' in the character-string is used to represent a character position which contains any allowable character from the computer's character set.

Z Each 'Z' in a character-string may only be used to represent the leftmost leading numeric character position which will be replaced by a space character when the contents of that character position is zero. Each 'Z' is counted in the size of the item. * Each '*' in a character-string may only be used to represent the leftmost leading numeric character position which will be replaced by an asterisk character when the contents of that character position is zero. Each '*' is counted in the size of the item.

9 Each '9' in the character-string represents a character position which contains a numeral and is counted in the size of the item.

'.' When the character '.' (period) appears in the character-string it is an editing symbol which represents the decimal point for alignment purposes and in addition, represents a character position into which the character '.' will be inserted. The character '.' is counted in the size of the item. The insertion character '.' must not be the last character in a SCREEN-PICTURE character-string. The character '.' may only appear once in the character string.

, When the DECIMAL-POINT IS COMMA phrase is specified, the character ',' (comma) is used instead of '.' (period).

Note that GCOS 7 allows you to create forms where the decimal point is either a full stop (.) or a comma (,). DKU terminals, however, will only accept a full stop as a decimal point. To use a comma in a numeric edited field (i.e. SPIC 99,99) the field must be declared without the NU attribute, but in this case there is no control by the terminal of the characters input and the program must cater for this situation.



+ - The symbols '+' and '-' are used as editing sign control symbols. When used, they represent the character position into which the editing sign control symbol will be placed. This must be either the leftmost or the rightmost character position. These symbols are mutually exclusive in any given character string. Each character used in the character-string is counted in determining the size of the data item.

cs The currency symbol in the character-string represents a character position into which a currency symbol is to be placed. The currency symbol in a character-string is represented by the currency sign. The currency symbol is counted in the size of the item.

Editing Rules

1. There are three types of insertion editing available. These are:

- a) Simple insertion
- b) Special insertion
- c) Fixed insertion

2. Only numeric edited data is subject to edition.

3. Special Insertion Editing

The '.' (period) (or the ',' (comma) if the DECIMAL POINT IS COMMA phrase is specified) is used as the insertion character. In addition to being an insertion character it also represents the decimal point for alignment purposes. The insertion character used for the actual decimal point is counted in the size of the item. The result of Special Insertion Editing is the appearance of the insertion character in the item in the same position as shown in the character-string.

4. Fixed Insertion Editing

The currency symbol and the editing sign control symbols '+' and '-' are the insertion characters. Only one currency symbol and only one of the editing sign control symbols can be used in a given SCREEN-PICTURE character-string. The symbol '+' or '-' when used, must be either the leftmost or rightmost character position to be counted in the size of the item. The currency symbol must be the leftmost character position to be counted in the size of the item except that it can be preceded by either a '+' or a '-' symbol. Fixed insertion editing results in the insertion character occupying the same character position in the edited item as it occupied in the SCREEN-PICTURE character-string. Editing sign control symbols produce the following results depending upon the value of the data item:



EDITING SYMBOL IN PICTURE STRING	RESULT IF DATA POSITIVE OR ZERO	RESULT IF DATA NEGATIVE
+	+	-
-	space	-

5. As a general rule, for a given screen-picture and for a given set of attributes a valid output character string will always be considered as a valid input.

The rules for deriving the program's view of a SCREEN PICTURE are:

1. If a fixed insertion '+' or '-' appears in a SCREEN-PICTURE, the generated PICTURE clause will contain a leading 'S'.
2. For each '9' in the SCREEN-PICTURE, the generated PICTURE will contain a '9'.
3. For each 'Z' in the SCREEN-PICTURE, the generated PICTURE will contain a '9'.
4. For each 'X' in the SCREEN-PICTURE, the generated PICTURE will contain a 'X'.
5. For the decimal point symbol in the SCREEN-PICTURE, the generated PICTURE will contain a 'V'.
6. For the fixed insertion currency symbol in the SCREEN-PICTURE, the generated PICTURE will contain nothing.
7. For each '(nn)' in the SCREEN-PICTURE, the generated PICTURE will contain a '(nn)'.
8. For each '*' in the SCREEN-PICTURE, the generated PICTURE will contain a '9'.



4.3.7 VALUE Clause

The VALUE clause is used to specify an initial value for a field on the screen.

General Format

VALUE IS literal

General Rules

1. Literals must be bounded by quotation marks ("). They may contain any character in the computer character set. If a quotation mark occurs within the literal, two contiguous quotation marks must be used to represent a single occurrence of the quotation mark.
2. If the VALUE clause is specified on an NFIELD entry, the VALUE clause specifies the initial value for the screen and is not treated as the initial value in the data record of the application program.
3. For UFIELDS, the VALUE clause may not be omitted. It specifies the fixed value of the field for the displayed form.
4. The maximum size that may be defined for the literal is 80 characters.

4.3.8 USAGE Clause

General Format

USAGE IS [COMP-1/COMP-2/COMP-3/COMP-8].

General Rules

1. The USAGE clause may be specified for NFIELD statements only.
2. When the USAGE clause is specified the field in the data record is generated as COMP-1 , COMP-2, COMP-3 or COMP-8.
3. A USAGE clause implies a numeric or numeric edited SPIC. For COMP-1 and COMP-2, the SPIC must not contain a decimal point.



4.3.9 FUNCTION Clause

The FUNCTION clause is used to define a field where the value of function keys is stored.

General Format

{FUNCTION-CODE | FC}

Syntax Rules

1. With the exception of the data-name clause, no other clauses may be specified with the FUNCTION clause.
2. The FUNCTION clause may only be specified in a NFIELD entry.
3. The FUNCTION clause may only be specified once in a form.

General Rules

1. The FUNCTION clause implies a PICTURE of XX.
2. The declaration of an explicit function code field prevents MAINTAIN FORM from generating the implicit function code field. This option is provided for compatibility with previous releases and it is recommended not to use it and to receive function keys in the implicit function code field.
3. During the execution of a receive, the value of the function key will be stored in the NFIELD described with the FUNCTION clause in the way described below.



4.4 ARRAY/END-ARRAY Statements

The ARRAY/END-ARRAY statements are used to define a repetitive group of fields in a form.

General Format

ARRAY data-name OCCURS integer TIMES.

{NFIELD / UFIELD statement}...

END-ARRAY.

General Rules

1. The ARRAY/END ARRAY statements define an array whose name is data-name and whose dimension is integer.
2. Several arrays may be defined in a form, but no ARRAY statement may appear within an array. That is nested arrays are not allowed.
3. The NFIELD and UFIELD statements between ARRAY and END ARRAY define the first element of the array. Specifically, their LINE clause refer to this first element. This element must be composed of an integer number of lines. The line or block of lines defined by these statements are repeated integer times in the screen.
4. NFIELD statements that are not part of the array cannot define fields located at a line belonging to the array.
5. UFIELD statements that are not part of the array can define fields located at a line belonging to the array provided that there is no overlapping between these non repeatable fields and the fields of the array. Such statements must follow the array definition in the form definition language.
6. Integer must be greater than 1 and less than or equal to 24.
7. When an ARRAY request is used and a form pictorial description is given, the first occurrence of the repetitive structure is defined explicitly in the form pictorial description. There must be as many blank lines as needed to display the expanded array.



4.5 Attribute Codes

The attributes of a field are supplied through a list of keywords.

They consist of the following:

- Rendition attributes that affect the visual aspect of a field: BI, BD, Bxxx, COS, CSAM, CSAP, CSPS, CN, DFT, FT, Fxxx, HL, H200, RV, UL, and W200. These may be specified for either NFs or UFs except CSAM which is accepted only for UFs, and CSAP which is accepted only for NFs.
- Qualification attributes that affect the entry of data in the field and may be specified only for NFs: AN, DT, DI, IT, RQ, IDBD, JL, JR, MF, NTR, NU, LP, PR, TR, NPR. Note that AN, JL, LP, TR and NPR are default values and that, when PR is specified, the other qualification attributes do not apply. The UF's are generated with the implicit attribute NLP (not printable); when a local print of UF's fields is requested, they must be generated explicitly with the attribute LP.
- A pseudo attribute: CP that applies only to unprotected NFs.

A description and meaning of each is supplied below.

ANY-CHARACTER (Abbreviation AN): any character may be entered into the field.

BACKGROUND-COLOR-mnemonic (Abbreviation Bxxx): the background is displayed in the color specified by the mnemonic.

Mnemonics:

RED Red
YEL Yellow
BLU Blue
GRE Green
CYA Cyan
MAG Magenta
WHI White
BLA Black
DFT Default color

BLINKING (Abbreviation BI): the field is blinking.

BOLD (Abbreviation BD): the rendition of the field is bold or increased intensity.



CHARACTER-SET-AM (Abbreviation CSAM): the character set of the field is alphamosaics. This attribute is supported only for UFIELDS. When this attribute is specified, other rendition attributes are ignored, except the color attributes (Bxxx and Fxxx), and under-lined (UL). In this case, under-lined is interpreted as "separated characters", and not under-lined is interpreted as "contiguous characters". The VALUE clause of the field must consist of alphanumeric characters which occupy the same position in the ASCII character set as the desired character occupies in the alphamosaic character set. CSAM applies only to the MINTEL terminals.

CHARACTER-SET-PS (Abbreviation CSPA): the character set of the field is pseudo-graphics. When specified, the VALUE clause of the field is limited to the following characters with this interpretation:

# lower left-hand corner	└
% vertical line	
& upper left-hand corner	┌
' right-hand intersection	├
) lower right-hand corner	┘
* horizontal line	—
+ low intersection	┴
, upper right-hand corner	┐
- left-hand intersection	┤
. high intersection	┤
/ cross	+

For named fields, this attribute is not compatible with:

- numeric (NU), digit (DI).
- numeric PICTURE USING.
- numeric clause USING (COMP-1, COMP-2, COMP-5, COMP-8).

For terminals which do not support CSPA character set, the CSPA attribute is ignored.



Under interactive commands (ICREATE or MODIFY), CSPA characters for unnamed fields (UF must be set by their explicit values without specifying CSPA attributes. Thus, UF with CSPA characters cannot be set interactively on terminals that do not support this character set. On such terminals, CSPA UF characters can only be specified under CREATE or COMPILE.

COLUMN-SEPARATOR (Abbreviation COS): the characters of the field display a vertical bar at the right of the character window.

CONCEALED (Abbreviation CN): the entered characters are not displayed on the screen (but are transmitted if the field is transmittable).

CURSOR-POSITION (Abbreviation CP): when specified, it will cause the cursor to be positioned to the leftmost character of the field.

DEFAULT (Abbreviation DFT): the characters are displayed in the default rendition of the terminal, i.e. the effects of any previous BI, BD, COS, CN, FT, RV, JL attributes are suppressed.

DETECTABLE (Abbreviation DT): this field is detectable through a light pen.

DIGIT (Abbreviation DI): only the numeric digits (zero through nine) may be entered in this field.

ENTRY-REQUIRED (Abbreviation RQ): some data must be entered in this field by the terminal operator. This means the terminal physically prevents skipping over such a field without entering a value. This attribute may not be applied to a field having the protect attribute.

FAINT (Abbreviation FT): the rendition of the field is reduced intensity.

BACKGROUND-COLOR-mnemonic (Abbreviation Fxxx): the characters are displayed in the color specified by the mnemonic.

Mnemonics:

RED Red
YEL Yellow
BLU Blue
GRE Green
CYA Cyan
MAG Magenta
WHI White
BLA Black
DFT Default color

HEIGHT-RATIO200 (Abbreviation H200): the characters in the field are to be of double height. This attribute is ignored if the corresponding positions on the line preceding the field are not empty, or if the field is on the first line. You cannot define fields on consecutive lines with this attribute.



For example,

$\begin{matrix} \triangle & A & \triangle & A \\ & A & & A \end{matrix}$

is allowed, but

$\begin{matrix} \triangle & & A & A \\ & & \triangle & A \\ A & & & A \end{matrix}$

is not allowed

HIGHLIGHT (Abbreviation HL): the rendition of this field is highlighted in a terminal-specific way. For each terminal type this attribute is mapped to a rendition attribute available on the terminal.

IMMEDIATE-TRANSMIT (Abbreviation IT): this field must also have the **DETECTABLE** attribute. When selected with the light pen, it causes an immediate transmission of the list of detected fields.

INPUT-DEVICE-BD (Abbreviation IDBD): when this attribute is used, data can only be entered from the badge reader.

JUSTIFY-LEFT (Abbreviation JL): field will be left-justified for the screen image and transmission to the host.

JUSTIFY-RIGHT (Abbreviation JR): the field will be right-justified for the screen image and transmission to the host.



CAUTION:

The JL and JR attributes will not work correctly unless the field is empty before any input from the keyboard. Note that, on certain terminals, a field may wrongly appear to be empty, whereas it is in fact filled with space characters. This can be the result of, for example, an initial value, an explicit send action, or a clear action on an protected field.

MUST-FILL (Abbreviation MF): if there is an entry in this field, it must occupy all the positions in the field.

NOT-PRINTABLE (Abbreviation NLP): the value of the field is not printed when a print is executed.

NOT-TRANSMITTABLE (Abbreviation NTR): the contents of the fields are never transmitted to the program.

NUMERIC (Abbreviation NU): only the numeric characters (zero through nine, currency sign, sign, decimal point) may be entered in this field.



PRINTABLE (Abbreviation LP): the value of the field will be printed when a print is executed.

PROTECTED (Abbreviation PR): the value of a protected field may not be altered by operator. This means the terminal physically prevents entry of data in such a field by other than the application program.

REVERSE-VIDEO (Abbreviation RV): field is in reverse video (Negative image). If color attributes are specified, the effect of reverse video is to exchange the background and foreground colors.

TRANSMITTABLE (Abbreviation TR): permits the transmission of the contents of a field.

UNDERLINED (Abbreviation UL): the field is underlined.

UNPROTECTED (Abbreviation NPR): the field is not protected.

WIDTH-RATIO200 (Abbreviation W200): the characters in the field are double width. This attribute is ignored if the field is not followed by a number of empty positions equal to the field length.



5. Terminal Dependent Features

Although MAINTAIN_FORM aims at terminal independence, a form designer should be aware of the limitations or peculiarities that pertain to the terminal types on which a form is to be used. The following paragraphs summarize these terminal specific features.

5.1 DKU7005/7

5.1.1 Limitations

The number of unprotected named fields on a screen is limited to 128. This limitation is checked by MAINTAIN_FORM at form level. However, when several forms are simultaneously active on a screen, you should check that the total number of unprotected fields of these forms does not exceed the limit.

Up to 15 rendition attributes may be defined on a line. Note that a field for which rendition attributes are specified (FT, UL, BI, RV, BD, HL,...) generates two attributes for the terminal: one to set the attributes and another one to reset them at the end of the field (except in the case of a field terminated in column 80 where the resetting is implicit). Moreover, MAINTAIN_FORM systematically generates a NORMAL attribute at the end of the first 7 named fields of a line in order to allow further dynamic attribute modification on these fields.

When a form is to be activated in OVERLAY mode or following a CDMECH PROTCT, it must contain at least one unprotected field. Otherwise, the previous forms will not be protected.

When a form contains an array generated with the RPLF functionality, the form is activated only at the top of the screen in Append mode.



5.1.2 Attributes

Available rendition attributes are BI, CN, UL, RV, and FT. HL is mapped to RV. BD is mapped to RV when the SUBSTITUTE ATTRIBUTE clause is selected. UFs with CSPA attribute are not generated. Other rendition attributes are ignored.

The rendition attributes occupy one position on the screen. MAINTAIN_FORM places the attributes at the positions preceding and following the concerned field. Therefore it is recommended that any field on which a static or dynamic rendition attribute is applied be preceded and followed by a space. If this is not the case, MAINTAIN_FORM may displace or truncate adjacent UFs.

The qualification attributes (beginning and end of an unprotected field) do not occupy a position on the screen. However, two consecutive unprotected fields must be separated by at least one character. The only exception to this rule is that an NF may be terminated on column 80 of a line and another one start on column 1 of next line.

In some cases, a space in a field may stop a rendition attribute. This limitation is removed by MAINTAIN_FORM and the SDPI access method by either replacing spaces by cursor forward positioning or regenerating the attribute after sending a field.

Available qualification attributes are: AN, DI, RQ, IDBD, JL, JR, MF, NU, LP, PR and NPR. All unprotected fields are transmittable and vice versa. Therefore, the TR attribute is ignored by MAINTAIN_FORM that generates a field from the PR/NPR attribute. If a form is to be used on DKU7007 and other terminals at the same time, it is recommended that PR fields are also declared as NTR.

At run time, if you want to use the attributes JL, JR, MF, RQ on a receive action, you must apply a clear on the keyboard onto the corresponding fields which are not empty.

5.1.3 RECEIVE REAL-MODE

When the RECEIVE REAL-MODE option is selected, all unprotected non empty fields are received, even if they are filled by spaces.



5.1.4 Function Key Mapping

When the implicit function key field is used, the most recent function key is mapped to a rank as follows:

- CTL1 to CTL9 are mapped to function codes 1 to 9 (either normal pad or numeric pad).
- CTL D to CTL R are mapped to function codes 10 to 24.

If an explicit function key field is defined, the FC1 FC2 characters of the VIP header are returned directly. If other function keys have been activated, they are mapped to function code 0.

5.1.5 Switches

Blink and blank codes are not used as rendition attributes. Therefore, it is recommended to deactivate the "blink and blank" mode through the corresponding switch (B1=0).

Selection switch A1 on DKU7007 (or C8 or DKU7005) must be set to 0.

5.1.6 Local Form Storage on Diskette

Local form storage on diskette is available with the following limits:

- The maximum number of data sets on a diskette is 14.
- The maximum number of forms per data set is 184.

In the terminal configuration utility, the O command must specify that the diskette does not work in VIP mode and codes returned in the frame header must not be redefined (that is, ACK must be returned as 61, PGOFF as 62, and NA as 63).

5.1.7 Character Sets

The DKU7005/7 terminal uses the 94 characters of set C101.



5.2 DKU7107

5.2.1 Limitations

The number of transmissible named fields on a screen is limited to 192.

This limitation is checked by `MAINTAIN_FORM` at form level. However, when several forms are simultaneously active on a screen, you should check that the cumulated number of transmissible fields of these forms does not exceed the limit.

When a form is to be activated in `OVERLAY` mode or following a `CDMECH PROTECT`, it must contain at least one unprotected field. Otherwise, the previous forms will not be protected.

When a form contains an array generated with the `RPLF` functionality, the form is activated only at the top of the screen in Append mode.

5.2.2 Attributes

Available rendition attributes are `BI`, `COS`, `CN`, `UL`, `RV`, `FT`, and `CSPS`. `HL` is mapped to `RV`. `BD` is mapped to `RV` when the `SUBSTITUTE ATTRIBUTE` clause is selected. Other rendition attributes are ignored.

The rendition attributes other than `CSPS` occupy one position on the screen. `MAINTAIN_FORM` places the attributes at the positions preceding and following the concerned field. Therefore it is recommended that any field on which a static or dynamic rendition attribute is applied be preceded and followed by a space. If this is not the case, `MAINTAIN_FORM` may displace or truncate adjacent UFs.

The qualification attributes (beginning and end of an unprotected field) do not occupy a position on the screen. However, two consecutive unprotected fields must be separated by at least one character. The only exception to this rule is that an `NF` may be terminated on column 80 of a line and another one start on column 1 of next line.

In some cases, a space in a field may stop a rendition attribute. This limitation is removed by `MAINTAIN_FORM` and the `SDPI` access method by either replacing spaces by cursor forward positioning or regenerating the attribute after sending a field.



Available qualification attributes are AN, DI, RQ, IDBD, JL, JR, MF, NU, LP, TR, NTR, PR, and NPR. All unprotected fields are transmittable but protected fields are also transmissible except if declared as NTR. Except for specific usage, and for performance reasons, it is recommended to declare all PR fields as NTR.

At run time, if you wish to use the attributes JL, JR, MF, RQ on a receive action, you must apply a CLEAR mechanism (or clear on the keyboard) onto the corresponding fields which are not empty.

5.2.3 RECEIVE REAL-MODE

When the RECEIVE REAL-MODE option is selected, all unprotected non-empty fields are received, even if they are filled by spaces.

5.2.4 Function Key Mapping

When the FKC function keys have not been locally mapped onto character strings, they may be received in the function code field. These may be transmitted either alone or with data, depending on a technical selection, both cases being supported. Function keys F1-F12 are mapped to function codes 01-12.

Moreover, VIP mode function keys are supported in the same way as for DKU7007.

5.2.5 Technical Selections

Blink and blank codes are not used as rendition attributes. It is therefore recommended to deactivate the blink and blank mode through the corresponding technical selection.

Emulation mode must be set to VIP7760 (not to VIP7700).

The attribute mode selection must be set to VIP mode and not to SDP mode.

5.2.6 Character sets

In addition to the 94 characters of set C101, the DKU7107 terminal uses set C127 for PLW (Pluri Lingual West with Western European accented characters), PLE (Pluri Lingual East with Eastern European accented characters), and sets C094, C113, C114, and C118 for Chinese, Cyrillic, Arabic, and Greek characters respectively. It also supports CSPS (pseudo-graphics).



5.3 DKU7211

5.3.1 Limitations

The number of transmissible named fields on a screen is limited to 192.

This limitation is checked by MAINTAIN_FORM at form level. However, when several forms are simultaneously active on a screen, the user should check that the cumulated number of transmissible fields of these forms does not exceed the limit.

When a form is to be activated in OVERLAY mode or following a CDMECH PROTECT, it must contain at least one unprotected field. Else, the previous forms will not be protected.

When a form contains an array generated with the RPLF functionality, the form is activated only at the top of the screen in Append mode.

5.3.2 Attributes

Available rendition attributes are Bxxx, CN, RV, Fxxx and CSPS. However, the terminal does not allow you to specify a background and a foreground color simultaneously for the same field since one of these colors is always black. When both a foreground and a background color attribute are specified for a field, the following rule applies: if RV is not specified, the Fxxx attribute is ignored; if RV is specified, the Bxxx attribute is ignored. HL is mapped to BRED. The HL attribute takes precedence over any other color attribute. Other rendition attributes are ignored.

The rendition attributes other than CSPS occupy one position on the screen. MAINTAIN_FORM places the attributes at the positions preceding and following the field concerned. Therefore it is recommended that any field on which a static or dynamic rendition attribute is applied be preceded and followed by a space. If this is not done, MAINTAIN_FORM may displace or truncate adjacent UFs.

The qualification attributes (beginning and end of an unprotected field) do not occupy a position on the screen. However, two consecutive unprotected fields must be separated by at least one character. The only exception to this rule is that an NF may be terminated on column 80 of a line and another one start on column 1 of next line.



In some cases, a space in a field may stop a rendition attribute. This limitation is removed by `MAINTAIN_FORM` and the `SDPI` access method by either replacing spaces by cursor forward positioning or regenerating the attribute after sending a field.

Available qualification attributes are `AN`, `DI`, `RQ`, `IDBD`, `JL`, `JR`, `MF`, `NU`, `LP`, `TR`, `NTR`, `PR`, and `NPR`. All unprotected fields are transmissible but protected fields are also transmissible except if declared as `NTR`. Except for specific usage, and for performance reasons, it is recommended to declare all `PR` fields as `NTR`.

At run time, if you wish to use the attributes `JL`, `JR`, `MF`, `RQ` on a receive action, you must apply a `CLEAR` mechanism (or clear on the keyboard) onto the corresponding fields which are not empty.

5.3.3 RECEIVE REAL-MODE

When the `RECEIVE REAL-MODE` option is selected, all unprotected non empty fields are received, even if they are filled by spaces.

5.3.4 Function Key Mapping

When the `FKC` function keys have not been locally mapped onto character strings, they may be received in the function code field. These may be transmitted either alone or with data, depending on a technical selection, both cases being supported. Function keys `F1-F12` are mapped to function codes `01-12`.

Moreover, `VIP` mode function keys are supported in the same way as for `DKU7007`.

5.3.5 Technical Selections

The color selection must be set to 7-color mode.

Blink and blank codes are not used as rendition attributes. Therefore, it is recommended to deactivate the blink and blank mode through the corresponding technical selection.

Emulation mode must be set to `VIP7760` (not to `VIP7700`).

The attribute mode selection must be set to `VIP` mode and not to `SDP` mode.



5.3.6 Character sets

In addition to the 94 characters of set C101, the DKU7211 terminal uses set C127 for PLW (Pluri Lingual West with Western European accented characters), PLE (Pluri Lingual East with Eastern European accented characters), and sets C094, C113, C114, and C118 for Chinese, Cyrillic, Arabic, and Greek characters respectively. It also supports CSPS (pseudo-graphics).



5.4 VIP7804/HDS7/VIP8800

5.4.1 Limitations

The number of attributes per line is limited to 16. This number may be calculated as follows:

Define each part of the line that is not occupied by a named or unnamed field as a space field. UFs which do not have a rendition attribute and an adjacent space field are concatenated to create a new UF.

Adjacent UFs with the same attributes are concatenated to create a new UF. This limitation applies to the VIP7804 but not to the VIP7814. When the limit is reached, a warning message is issued, but the form generation is completed.

Each NF, UF, and space field requires one VIP7804 attribute field.

HDS7 and VIP8800 Only:

The parameter TERM = ANY must be specified for each command that accepts the parameter TERM.

5.4.2 Attributes

Available rendition attributes are BI, CN, UL, RV, FT, and CSPS. HL is mapped to RV. BD is mapped to RV when the SUBSTITUTE ATTRIBUTE clause is selected. Other rendition attributes are ignored.

Available qualification attributes are: AN, DI, RQ, JL, JR, MF, NTR, NU, LP, PR, TR, and NPR. All unprotected fields are transmissible but protected fields are also transmissible except if declared as NTR.

The attributes do not occupy any position on the screen. However, a position where an attribute is set cannot contain a pseudo-graphic character. Therefore, any UF with CSPS attribute should be preceded by a space and, in the NFs, the first CSPS character will always be a space.

When CSPS is used with CN, CN is ignored.

NOTE:

CSPS characters may be entered with the FAINT (FT) and/or blink (BK) attributes. If a field has the FT and/or BK attributes, it is advisable to enter the CSPS characters with the corresponding attribute. Otherwise, these attributes (FT and/or BK) will not be taken into account.



5.4.3 RECEIVE REAL-MODE

The RECEIVE REAL-MODE clause has no effect on VIP7804/HDS7/VIP8800 terminals. All transmittable non empty fields are received.

5.4.4 Function Key Mapping

Function keys F1-F12 are mapped to function codes 1-12.

Shifted function keys F1-F12 are mapped to function codes 13-24.

VIP7804 Only:

When you press a function key the code is transmitted immediately (data entered on the screen from the keyboard is not sent). FORMS refreshes the screen with the previous data immediately after receiving the function key in order to erase data input from the keyboard but not received by GCOS 7.

5.4.5 72-Line Option

Although a form may not exceed 24 lines, you can take advantage of the 72-line option by using the APPEND mode of the CDGET primitive at run time. Several forms may be appended, provided that they do not exceed the declared page length. When this feature is to be used, you should check that the terminal has been declared with a 72-line page length at network generation.

5.4.6 Character sets

In addition to the 94 characters of set C101, the VIP7804/HDS7/VIP8800 terminals also supports CSPS (pseudo-graphics).

VIP8800 Only:

The VIP8800 terminal (8-bit terminal) uses set C127 (with Western European accented characters) and set C118 for Greek characters.



5.5 PC7800

IBM/PC and compatible personal computers running MS-DOS can be connected to a DPS7 via the PC7800 asynchronous terminal emulator.

The terminal PC7800 is added to the list of terminals supported by FORMS. Therefore TERM = 7800 may be specified for each command that accepts the parameter TERM.

5.5.1 Limitations

There is no specific limitation on the number of fields or attributes.

5.5.2 Attributes

- Available rendition attributes are BI, CN, UL, RV, FT, Bxxx, Fxxx, and CSPA. HL is mapped to BRED. BD mapped to RV when the SUBSTITUTE ATTRIBUTE clause is selected. Other rendition attributes are ignored.
- The UL attribute is incompatible with RV. If both are specified for a field, UL is ignored. Moreover, UL is never available in the case of a color terminal.
- Fields with RV or Bxxx attributes cannot be displayed in normal intensity. They are always, displayed with FT attribute.
- Available qualification attributes are: AN, DI, RQ, JL, JR, MF, NTR, NU, LP, PR, TR, and NPR. All unprotected fields are transmittable but protected fields are also transmittable unless declared as NTR.
- When the CSPA attribute is used with CN, CN is ignored.

NOTE:

CSPA characters may be entered with the faint (FT) and/or blink (BK) attributes. If a field has the FT and/or BK attributes, it is advisable to enter the CSPA characters with the corresponding attribute. Otherwise, these attributes (FT and/or BK) will not be taken into account.

5.5.3 RECEIVE REAL-MODE

The RECEIVE REAL-MODE clause has no effect on PC7800. All transmittable non empty or space filled fields are received.



5.5.4 Function Key Mapping

Function keys F1-F10 are mapped to function codes 1-10.

Function keys F1-F2 combined to CTL key are mapped to function codes 11-12.

Shifted function keys F1-F10 are mapped to function codes 13-22.

Function keys F1-F2 combined to ALT key are mapped to function codes 23-24.

5.5.5 Configuration

In order to connect the PC7800 to GCOS 7, you must modify the configuration file. To do this, you must first load the PC7800 software.

Whenever the VIP7800 series screen format is displayed, enter the command "ALT S".

Once you have issued an ALT S command, the following menu will be displayed on your screen:

Configuration Menu:

```
(Q) Quit or return to exit
(C) Color Change
(B) Baud Rate Change
(R) Rear Panel Switch
(X) Extended Configuration Menu
(S) Save Configuration
(L) Load a Configuration File
```

Select an item--

You should issue the R command that will display a sub-menu. This sub-menu should be set to the following contents.



Rear Panel Switch Menu:

```
Q or return to exit this menu
A Parity          (0 = odd; 1 = even)      1
B Parity enabled                                     1
C CR/LF in text mode                               1
D CHAR/TEXT      (0 = TEXT; 1 = CHAR)      0
E Space suppress                                     0
F Message term. (0 = ETX; 1 = EOT)          0
G Echoplex mode (Tx-ret if text mode)       0
H Roll mode                                          1
I Hold DTR in local mode                       0
J Display all characters                         0
K Autotab (0 = delayed; 1 = immediate)       1
L Transmit (0 = nonblock; 1 = block mode)    0
M Cursor (0 = underline; 1 = block )        0
```

Select an item--

Issuing the X command of the Configuration Menu will give you the following Extended Configuration Menu.

This sub-menu is used to define file transfer parameters (MICROFIT7-MIMIC).

Extended Configuration Menu

```
Q or return to exit this menu
A Path name for GCOS/GCOS8 protocol transfer FTRAN
B Set character size (7 or 8 bits)             7
C # seconds after each line in Opt. C file transfer 0
D communications port to use (0 or 1)         0
E File transfer - send packet size            1024
F File transfer - receive packet size         1024
```

Select an item--



5.5.6 Use of the Keyboard

PC7800 provides a help menu which you can access at any time during a terminal session by entering the command ALT H. This causes the screen to be saved and the help menu to be displayed. This menu indicates which keys should be pressed to perform local functions.

The keyboard may be set in pseudo-graphics mode by pressing the ESC key followed by an uppercase G. Thereafter, pseudo-graphics may be entered by pressing A to K uppercase keys. To exit from graphics mode, press the ESC key followed by an uppercase F.

The messages which are sent to the Datanet or to GCOS 7 must be terminated by the '+' key on the right of the keyboard.

5.5.7 Character sets

The PC7800 terminal uses the 94 characters of set C101.



5.6 IBM3270

5.6.1 Limitations

There is no specific limitation on the number of fields or attributes.

5.6.2 Attributes

The only available rendition attribute is BD. HL is mapped to BD. RV and BI are mapped to BD when the SUBSTITUTE ATTRIBUTE option is selected. UFs with CSPA attribute are not generated. Other rendition attributes are ignored.

Each named field is preceded and followed by an attribute that generates both the appropriate qualification and rendition. This attribute occupies one character on the screen. Therefore each named field should be preceded and followed by a space.

A named field with DT attribute must be preceded by at least 4 spaces.

Available qualification attributes are: AN, DI, DT, IT, NTR, NU, PR, TR and NPR. All unprotected fields are transmissible but protected fields are also transmittable except if declared as NTR. The attribute DI is mapped in NU.

5.6.3 RECEIVE REAL-MODE

When the RECEIVE REAL-MODE option is selected, only the modified fields are transmitted. Else, although the terminal actually transmits only the modified fields, the program gets the visibility of all non-empty and non-space-filled fields being transmitted.

5.6.4 Function Key Mapping

Function keys PF1 to PF24 are mapped to function codes 1 to 24 when received in a function code field (either explicit or implicit).

5.6.5 Character sets

The IBM3270 terminal uses the 94 characters of set C101.



5.7 IBM3278 - IBM3279

5.7.1 Supported Configuration

There is no standard profile for each IBM terminal which gives exactly all its capabilities. There are:

- five IBM3278 models (2, 2A, 3, 4, and 5)
- six IBM3279 models (S2A, S2B, S3, 2C, 02X, and 03X)
- two controllers: 3274 and 3276.

The number of possible combinations among these components does not allow a specific adaptation for each. It is therefore impossible to be sure that the SDPI/IBM interface can run exactly for all IBM configurations. Consequently, when possible, FORMS uses the implicit IBM folds.

FORMS supports only two IBM profiles:

IBM3270-like

IBM3278/3279 with:

extended highlighting - colors.

The size of the screen supported is the standard size used by GCOS 7: 24 lines by 80 columns.

5.7.2 Limitations

There is no specific limitation on the number of fields or attributes.



5.7.3 Attributes

IBM3278/3279 supports the field attributes which are the same as for IBM3270:

- protected/non-protected
- numeric/non-numeric
- concealed, non-printable, non-detectable
- concealed, normal intensity, detectable
- bold, detectable
- normal, non-detectable.

It also supports extended attributes, which are:

- reverse video
- blink
- CSAP
- CSPS.

The attribute DI is mapped to NU.

For IBM3279, the color attributes are supported: Bxxx and Fxxx. However, the terminal does not allow you to specify a background and a foreground color simultaneously for the same field. When both a foreground and a background are specified for a field, the following rules apply: if RV is not specified, the Fxxx attribute is ignored; if RV is specified, the Bxxx attribute is ignored. HL is mapped to BRED. The HL attribute takes precedence over any other color attribute.

Independently of MAINTAIN_FORM, the 3279 terminals model 2A/3A and 2B/3B give an implicit correspondence between qualification attributes and colors. A local switch (base mode color) is used to pass from monochrome to color mode.

FIELD ATTRIBUTE	MONOCOLOR	COLOR
UNPROTECTED NORMAL INTENSITY	Green	Green
UNPROTECTED INTENSIFIED	White	Red
PROTECTED NORMAL INTENSITY	Green	Blue
PROTECTED INTENSIFIED	White	White



5.7.4 Color Attribute 3279 2B/3B

You may choose among eight colors for foreground and for background. The correspondence SDPI/IBM for those colors is described in this table:

FORMS COLOR	3278	3279
Default	green	green
Blue	ignored	blue
Red	ignored	red
Magenta	ignored	pink
Green	ignored	green
Cyan	ignored	turquoise
Yellow	ignored	yellow
Black	ignored	black
White	ignored	white

Resolution of Attributes Conflicts

IBM3278/3279 support the following extended attributes:

- HL: Highlight
- RV: Reverse-Video
- UL: Underline
- BK: Blink












Only one of them may be displayed at a time. Priorities have been defined: HL > RV > UL > BK.



5.7.5 Character Sets

Two character sets are supported by IBM3278/3279 for use with Forms. These are the 94-character set C101, and the PLW character set C127 which provides the western European accented characters. When the PLW set is active, Forms does not use a transcoding table and reproduces the data exactly as it is entered.

The CSPS (pseudo-graphic) characters are defined using a 2-byte interface). They correspond to the 12 characters previously supported on DKU7211, DKU7107, and VIP7804/HDS7/VIP8800 terminals. These characters are presented below, with the corresponding DKU characters:

	IBM	DKU	REPRESENTATION	
#	"C4"X	"50"X	lower left-hand corner	
%	"BF"X	"6C"X	vertical line	
&	"C5"X	"50"X	upper left-hand corner	
'	"C6"X	"7D"X	right-hand intersection	
)	"D4"X	"5D"X	lower right-hand corner	
*	"A2"X	"5C"X	horizontal line	
+	"C7"X	"4E"X	low intersection	
'	"D5"X	"6B"X	upper right-hand corner	
-	"D6"X	"60"X	left-hand intersection	
.	"D7"X	"4B"X	high intersection	
/	"D3"X	"61"X	cross	
line	"40"X	"40"X	space	

The representation of IBM pseudo-graphic characters requires lowercase letters.



5.7.6 RECEIVE REAL-MODE

When the RECEIVE REAL-MODE option is selected, only the modified fields are transmitted. Otherwise, although the terminal actually transmits the modified fields, the program keeps the visibility of all non-empty and non-space-filled fields being transmitted.

5.7.7 Function Key Mapping

Function keys PF1 to PF24 are mapped to function codes 1 to 24 when received in a function code field (either explicit or implicit). The correspondence is as follows:

IBM KEY	FORMS RANK
F1 through F9	1 through 9
7A, 7B, 7C	10, 11, 12
C1 through C9	13 through 21
4A, 4B, 4C	22, 23, 24

5.7.8 Light Pen Processing

To use the light pen, you must define the variable field with the DETECTABLE (DT) or with the two attributes DETECTABLE and IMMEDIATE_TRANSMIT (IT). A named field with a DT or DT + IT attribute(s) must be preceded by at least 4 spaces. Two cases are possible, which are described below. In either case, when data is received in the field, the corresponding selection vector entry is set to "D" (except when the CPON mechanism is activated, for it takes precedence). A maximum of 15 fields with DT or IT attributes may be defined.

- Field defined with the DT attribute: the first character of the field is "?". When selected, the character "?" is translated to "<" indicating that the field has been modified. The field may be deselected by using the same technique as for selection. After a receive, the "?" remains in the data record if the field is not selected by the light pen.
- Field defined with the DT and IT attributes: the first character of the field is "&".



5.8 MINITEL (40 Columns)

The alphamosaic character set given in the following table applies only for the unnamed fields (UF) on the MINITEL terminal. It is a repertoire of characters which can be used to produce boxes or frames in a form description or histogram or similar diagram (an example is given below).

5.8.1 Alphamosaic Character Set

Table 5-1. Alphamosaic Character Set (1/2)

A M	CODES			A M	CODES			A M	CODES			A M	CODES		
	A	E	C		A	E	C		A	E	C		A	E	C
00 00 00	20	40 ▽	65	00 00 X0	30	F0 0	241	00 00 0X	60	79	122	00 00 XX	70	97 P	152
X0 00 00	21	4F '	80	X0 00 X0	31	F1 1	242	X0 00 0X	61	81 a	130	X0 00 XX	71	98 q	153
0X 00 00	22	7F "	128	0X 00 X0	32	F2 2	243	0X 00 0X	62	82 b	131	0X 00 XX	72	99 r	154
XX 00 00	23	7B #	124	XX 00 X0	33	F3 3	244	XX 00 X0	63	83 c	132	XX 00 XX	73	A2 s	163
00 X0 00	24	5B \$	92	00 X0 X0	34	F4 4	245	00 X0 0X	64	84 d	133	00 X0 XX	74	A3 t	164
X0 X0 00	25	6C %	109	X0 X0 X0	35	F5 5	246	X0 X0 0X	65	85 e	143	X0 X0 XX	75	A4 u	165
0X X0 00	26	50 &	81	0X X0 X0	36	F6 6	247	0X X0 0X	66	86 f	135	0X X0 XX	76	A5 v	166
XX X0 00	27	7D ,	126	XX X0 X0	37	F7 7	248	XX X0 0X	67	87 g	136	XX X0 XX	77	A6 w	167



Table 5-1. Alphamosaic Character Set (2/2)

A M	CODES			A M	CODES			A M	CODES			A M	CODES		
	A	E	C		A	E	C		A	E	C		A	E	C
00 0X 00	28	4D (78	00 0X X0	38	F8 8	249	00 0X 0X	68	88 h	137	00 0X XX	78	A7 x	168
X0 0X 00	29	5D)	94	X0 0X X0	39	F9 9	250	X0 0X 0X	69	89 i	138	X0 0X XX	79	A8 y	169
0X 0X 00	2A	5C *	93	0X 0X X0	3A	7A :	123	0X 0X 0X	6A	91 j	146	0X 0X XX	7A	A9 z	170
XX 0X 00	2B	4E +	79	XX 0X X0	3B	5E ;	95	XX 0X 0X	6B	92 k	147	XX 0X XX	7B	C0 {	193
00 XX 00	2C	6B '	108	00 XX X0	3C	4C <	77	00 XX 0X	6C	93 l	148	00 XX XX	7C	6A 	107
X0 XX 00	2D	60 -	97	X0 XX X0	3D	7E =	127	X0 XX 0X	6D	94 m	149	X0 XX XX	7D	D0 }	209
0X XX 00	2E	4B .	76	0X XX X0	3E	6E >	111	0X XX 0X	6E	95 n	150	0X XX XX	7E	A1 ~	162
XX XX 00	2F	61 /	98	XX XX X0	3F	6F ?	112	XX XX 0X	6F	96 o	151	XX XX XX	5F	6D _	110

Where:

A = ASCII (HEXA)
 E = EBCDIC (HEXA)
 C = COBOL
 AM = ALPHAMOSAICS



5.8.4 Attributes

Available rendition attributes are Bxxx, CN, RV, Fxxx, W200, H200, CSAM, UL, and BI. HL is mapped to BRED. The other rendition attributes are ignored (FT, CSPA).

The rendition attributes RV, UL, Bxxx, HL must be followed by a space. Therefore, those attributes occupy one position on the screen or two positions when the field is defined with W200 attribute.

MAINTAIN_FORM places the attributes one position before the field. In case of a field starting at column 1, MAINTAIN_FORM does not generate the attributes.

The qualification attributes do not exist on MINITEL. Therefore, they are simulated by SDPI. The available qualification attributes are: AN, NU, DI, PR, MF, RQ.

5.8.5 RECEIVE REAL-MODE

This option has no effect on MINITEL.

5.8.6 Function Keys

Since the MINITEL is not a block mode terminal, the function keys are used to navigate between the fields of the form which are simultaneously present on the screen. In other circumstances they can also be returned to the program, in which case they are always returned as a rank, no matter whether the function code field is implicit or explicit.

The following keys are used to navigate between fields:

- **RETOUR**
This key returns control to the program only if the current field is the first field on the screen. Otherwise, it sets the cursor to the previous field. When a "RETOUR" (return) key is used with the request attribute, characters must be re-entered in the field.
- **SUITE**
This key returns control to the program only if the current field is the last field on the screen. Otherwise, it sets the cursor to the next field.
- **REPETITION**
This key causes a logical restore of the screen contents. As it is possible to enter characters beyond the limits of named fields, the screen may display an inconsistent image. In such a case, the REPETITION key may be used to restore the screen to a consistent state.



- ENVOI
This key is the normal way to return control to the program. It plays the role of the TRANSMIT key in block mode terminals.
- ANNULATION
This key returns control to the program if the cursor is on the first character of a field. Otherwise, the cursor is set to the first character of the current field.

Other keys cause the control to be returned to the program with the function code field being loaded with a rank, according to the table below:

01	RETOUR (from first field only)
02	GUIDE
03	SOMMAIRE
04	SUITE (from last field only)
05	ANNULATION (from the first character of a field only)
06	CORRECTION (if the cursor was on the first character of a field; otherwise the key is directly handled by the VIDEOPAD)

5.8.7 Initial Values

The initial values of the named fields are displayed and retransmitted to the program on a CDRECV verb.



5.9 MINITEL 1B (Mixed Mode 40/80 Columns)

On initial connection, the Minitel 1B is in 40 column mode.

5.9.1 Switching between 40 column and 80 column mode

In IOF or for a TDS in Terminal Adapter Mode, you use the following command to select 40 or 80 columns:

```
MODIFY_PROFILE PW = 80|40
```

If the switching is done by application, use the CDMECH verb with:

```
TL80ON | TL80OFF
```

5.9.2 40 Column Mode

In 40 column mode, the Minitel 1B performs in all respects as the basic Minitel described in the previous sub-section.

5.9.3 80 Column Mode

In 80 column mode, the Minitel 1B uses the PLW character set (C127), giving access to the western European accented characters.

5.9.4 Limitations

For 80 column mode to be available, the ICREATE command must have been executed with TERM = ANY.



6. Examples Using Menus

This section consists of a sequence of menu screens showing how to create (Figures 6-1 through 6-11), test (Figures 6-12 and 6-13), and modify (Figures 6-14 through 6-34) a form.

6.1 ICREATE Examples

6.1.1 ICREATE Menu: A Form Named SALES is to be Defined

1/1 ICREATE -->:

create a new form interactively

FORM NAME + name of the form SALES

TERM ALL or type(s) of terminal(s)

LANG language used

The following parameter is meaningful only if LANG=DD

RECORD record name



6.1.2 Specification Phase

```

                ICREATE   SALES                -->:

NAMED FIELD CHARACTER   :                               #
UNNAMED FIELD STARTING CHARACTER :                       <
UNNAMED FIELD ENDING CHARACTER  :                       >
CURRENCY SIGN           :                               $
DECIMAL POINT           : (. or ,)                     .
FIELD TERMINATION CHARACTER: (VIP7700/7701 only)        .
RECEIVE SPACES: (0/1)                                     0
ALLOW RPLF: (0/1) (DKU7007,7107, 7211 only)           0
RECEIVE FUNCTION KEY: (0/1)                             1
DESCRIPTION:
    
```

6.1.3 Drawing Phase

```

#                <Product type> #####

<Name of representative:> ##### <Code:> #####

<----->
<|><Product<|> <Article><|><Quantity<|>< Price> <|> <Total> <|>
<|>                <|>                <|>                <|>                <|>
<|>##### <|>    ##### <|>    ### <|>    ##### <|>    ##### <|>

<----->
<----->
<|>
<|> < GRAND TOTAL : > ##### <|>
<----->
    
```




6.1.4 ENTER NF Request

```

                                ICREATE  SALES
-----1-----2-----3-----4-----5-----6-----

#           <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>

-----1-----2-----3-----4-----5-----6-----
-->: 1           NAMED FIELD (NF)           LINE: 04           DISPLAY FROM LINE: 01
           NF-RANK: 01

1:NF   4:RTV           NAME: SLASH

2:UF   5:TEST         ATTR: ?           COMP:
3:AR   6:MAP         SPIC: X(1)           INITIAL:

7:QUIT

```

An ENTER NF should be made for each NF whose parameters are different from default values. NF's for which no ENTER NF is made, are given an alphanumeric picture, default attributes and an automatically generated name (form-name-*nnn*).

If a question mark is entered in the first attribute field, an attribute definition form is prompted (see 6.1.5).



6.1.5 Attribute Definition Form

ATTRIBUTE LIST FOR NAMED FIELD

```

Protected  PR: 0   Numeric          NU: 0   Concealed      CN: 0
Printable  PT: 1   Blinking          BI: 0   Transmittable  TR: 1
Underlined UL: 0   Justify-right    JR: 0   Reverse-video  RV: 0
Digit      DI: 0   Faint            FT: 0   Must-fill      MF: 0
Detectable DT: 0   Entry-required   RQ: 0   Immediate-transmit IT: 0
Bold       BD: 0   Cursor-position  CP: 0   Column-separator COS: 0
Highlight  HL: 0   Height-ratio     H200: 0 Width-ratio     W200: 0
           (0:No, 1:Yes)

Input-device : 0
           (0:No, 1:Badge, 2:Document)

           APL          CSAP: 0
           Pseudo-graphics CSPS: 0

Foreground-colour      Fxxx: 0      Background-colour      Bxxx: 1
(Colour: 0:default DFT , 1:Red RED , 2:Yellow YEL ,
3:Blue BLU , 4:Green GRE , 5:Cyan CYA ,
6:Magenta MAG , 7:White WHI , 8:Black BLA )

```



6.1.6 ENTER UF Request

```

                                ICREATE   SALES
-----1-----2-----3-----4-----5-----6-----

#           <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>   ##### <.>   ### <.>   ##### <.>   ##### <.>

-----1-----2-----3-----4-----5-----6-----
-->: 2           NAMED FIELD (NF)           LINE: 04           DISPLAY FROM LINE: 01
           NF-RANK: 01

1:NF   4:RTV
2:UF   5:TEST           ATTR: FGRE
3:AR   6:MAP
       7:QUIT

```

An ENTER UF should be made for any UF enclosed within UF characters, in order to specify the rendition attributes that apply to this UF.



6.1.7 ENTER ARRAY Request

```

-----1-----2-----3-----4-----5-----6-----
                                ICREATE   SALES

#                               <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>

-----1-----2-----3-----4-----5-----6-----
-->: 3                                DISPLAY FROM LINE: 01

                                ARRAY   (AR)

1:NF   4:RTV                       STARTING-LINE: 12
2:UF   5:TEST                       NUMBER-OF-LINES: 01          DIMENSION: 07

3:AR   6:MAP                       ARRAY-NAME: SALE-LINES
7:QUIT                                DUPLICATE UNNAMED-FIELDS (1:Yes, 0:No) 1

```

An ENTER ARRAY should be made for each array in order to expand it.



6.1.8 RETRIEVE Requested

```

                                ICREATE   SALES
-----1-----2-----3-----4-----5-----6-----

#                               <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>

-----1-----2-----3-----4-----5-----6-----
-->: 4          NAMED FIELD (NF)          LINE: 04          DISPLAY FROM LINE: 01
          NF-RANK: 01

1:NF   4:RTV          NAME:
2:UF   5:TEST        ATTR:          COMP:
3:AR   6:MAP   SPIC:          INITIAL:
      7:QUIT

```

You specify here the line and rank of the NAMED-FIELD to be retrieved.

The RETRIEVE request retrieves information relative to a specified field (or array) as follows:

- an UF is referenced by (line, rank)
- an NF either by name or by (line, rank)
- an array by name or by its starting line.



6.1.9 RETRIEVE Request

```

                                ICREATE  SALES
-----1-----2-----3-----4-----5-----6-----

#           <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>

-----1-----2-----3-----4-----5-----6-----
-->: 4           NAMED FIELD (NF)           LINE: 04           DISPLAY FROM LINE: 01
                                NF-RANK: 01

1:NF   4:RTV           NAME: SLASH
2:UF   5:TEST         ATTR: BRED           CO
MP:
3:AR   6:MAP   SPIC: X(1)           INITIAL:

7:QUIT

```



6.1.10 TEST Request

Product type
Name of representative: Smith code: 1

```
.....  
. Product . Article . Quantity . Price . Total .  
. . . . .  
.....  
. drive . 65 . 150 . 15.00 . .  
. hook . 84 . 100 . 18.00 . .  
. swink . 51 . 200 . 16.00 . .  
. . . . .  
. . . . .  
.....  
.....  
. GRAND TOTAL : .  
.....
```

The TEST request shows the form being generated on the screen.

Pressing the TRANSMIT key will restore the screen to its previous contents.



6.1.11 QUIT Request

```

                                ICREATE   SALES
-----1-----2-----3-----4-----5-----6-----

#           <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ###  <.>  ##### <.>  ##### <.>

-----1-----2-----3-----4-----5-----6-----

Do you want to save ?           0
(1:Yes   0:No)
(2:Return to last request)
    
```

The QUIT request completes the command. You are asked whether you want to save the created form.

Note that the MAP request is irrelevant during ICREATE command processing.



6.1.15 Specification Phase

	MODIFY	SALES	---	:
NAMED FIELD CHARACTER :			#	
UNNAMED FIELD STARTING CHARACTER :			<	
UNNAMED FIELD ENDING CHARACTER :			>	
CURRENCY SIGN :			\$	
DECIMAL POINT : (. or ,)			.	
FIELD TERMINATION CHARACTER: (VIP7700/7701 only)			.	
RECEIVE SPACES: (0/1)			0	
ALLOW RPLF: (0/1) (DKU7007,7107, 7211 only)			0	
RECEIVE FUNCTION KEY : (0/1)			1	
DESCRIPTION:				



6.1.16 Drawing Phase

```

#                               <Product type> #####

<Name of representative:> ##### <Code:> #####

<----->
<|><Product<|><Quantity><|><Article><|>< Price> <|> <Total> <|>
<|>          <|>          <|>          <|>          <|>          <|>
<|>##### <|>   ### <|>   #### <|>   ##### <|>   ##### <|>

<----->
<----->
<|>
<|> < GRAND TOTAL : > ##### <|>
<----->

```

The form to be modified is prompted.



6.1.17 Drawing Phase - Modification

```
#                <Product type> #####

<Name of representative:> ##### <Code:> #####

<----->
<|><Product<|> <Article><|><Quantity<|>< Price> <|> <Total> <|>
<|>          <|>          <|>          <|>          <|>          <|>
<|>          <|>          <|>          <|>          <|>          <|>
<|>##### <|>   ##### <|>   ### <|>   ##### <|>   ##### <|>

<----->
<----->
<|>
<|> < GRAND TOTAL : > ##### <|>
<----->
```

The form is modified: line spaces are introduced and ARTICLE and QUANTITY are inverted.



6.1.18 Description Phase

```

-----1-----2-----3-----4-----5-----6-----
                MODIFY   SALES

```

```

#                <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>

```

```

-----1-----2-----3-----4-----5-----6-----
-->: 1                NAMED FIELD (NF)                LINE: 01                DISPLAY FROM LINE: 01
                NF-RANK: 01
1:NF   4:RTV
2:UF   5:TEST                NAME:
3:AR   6:MAP   SPIC:                ATTR:                COMP:
      7:QUIT                INITIAL:

```



6.1.19 Description Phase - Error

```

                                MODIFY   SALES
-----1-----2-----3-----4-----5-----6-----
#           <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>

-----1-----2-----3-----4-----5-----6-----
-->: a           NAMED FIELD (NF)           LINE: 01           DISPLAY FROM LINE: 01
1:NF   4:RTV
2:UF   5:TEST           NAME:
3:AR   6:MAP   SPIC:           ATTR:           COMP:
7:QUIT           INITIAL:

```

*UNKNOWN ACTION

If an error is detected, an error message is displayed on the last line.



6.1.20 Description Phase - Array Definition Request

```

-----1-----2-----3-----4-----5-----6-----
                                MODIFY   SALES

```

```

#                               <Product type> #####
<Name of representative:> ##### <Code:> #####
<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>

```

```

-----1-----2-----3-----4-----5-----6-----
-->: 3                                DISPLAY FROM LINE: 01
      NAMED FIELD (NF)                LINE: 01                NF-RANK: 01
1:NF   4:RTV                          NAME:
2:UF   5:TEST                          ATTR:                COMP:
3:AR   6:MAP   SPIC:                    INITIAL:
      7:QUIT

```

Array definition form is requested.



6.1.21 Description Phase - Array Definition

```

                                MODIFY   SALES
-----1-----2-----3-----4-----5-----6-----

#           <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ###  <.>    ##### <.>    ##### <.>

-----1-----2-----3-----4-----5-----6-----
-->: 4                                     DISPLAY FROM LINE: 01

                ARRAY   (AR)
1:NF   4:RTV                STARTING-LINE:   01
2:UF   5:TEST                NUMBER-OF-LINES: 01      DIMENSION:   02
3:AR   6:MAP                ARRAY-NAME:   Sale-lines
7:QUIT                        DUPLICATE UNNAMED-FIELDS (1:Yes, 0:No) 1
    
```

Array description form is prompted.
 A retrieve of SALE LINES parameters is requested.



6.1.22 Description Phase - RETRIEVE Request

```

                                MODIFY   SALES
-----1-----2-----3-----4-----5-----6-----
#                               <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>

-----1-----2-----3-----4-----5-----6-----
-->: 4                                     DISPLAY FROM LINE: 01

                                ARRAY   (AR)
1:NF   4:RTV                          STARTING-LINE:   00
2:UF   5:TEST                          NUMBER-OF-LINES: 01          DIMENSION:   02
3:AR   6:MAP                            ARRAY-NAME:   SALE-LINES
7:QUIT                                DUPLICATE UNNAMED-FIELDS (1:Yes, 0:No) 1

```

Result of the RETRIEVE request



6.1.23 Description Phase - Array definition

```

                                MODIFY   SALES
-----1-----2-----3-----4-----5-----6-----

#           <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>

-----1-----2-----3-----4-----5-----6-----
-->: 3                                     DISPLAY FROM LINE: 01

          ARRAY   (AR)
1:NF   4:RTV      STARTING-LINE:   12
2:UF   5:TEST     NUMBER-OF-LINES: 01      DIMENSION: 07
3:AR   6:MAP      ARRAY-NAME:  SALE-LINES
7:QUIT                                     DUPLICATE UNNAMED-FIELDS (1:Yes, 0:No) 1
    
```

New location of the array is introduced.



6.1.24 Description Phase - NF Request

```

                                MODIFY   SALES
-----1-----2-----3-----4-----5-----6-----

#           <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>

-----1-----2-----3-----4-----5-----6-----
-->: 1                                     DISPLAY FROM LINE: 01

          ARRAY   (AR)
1:NF    4:RTV      STARTING-LINE:   01
2:UF    5:TEST     NUMBER-OF-LINES: 01          DIMENSION: 02
3:AR    6:MAP      ARRAY-NAME:
7:QUIT                                DUPLICATE UNNAMED-FIELDS (1:Yes, 0:No) 1

```

Named field definition form is requested.



6.1.25 Description Phase - RETRIEVE Request

```

                                MODIFY   SALES
-----1-----2-----3-----4-----5-----6-----
#                               <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>

-----1-----2-----3-----4-----5-----6-----
-->: 4                                DISPLAY FROM LINE: 01
      NAMED FIELD (NF)                LINE: 01                NF-RANK: 01
1:NF   4:RTV                          NAME: quantity
2:UF   5:TEST                          ATTR:                COMP:
3:AR   6:MAP   SPIC: X(1)              INITIAL
      7:QUIT

```

Named field description form is prompted. A retrieve of QUANTITY field parameters is requested.



6.1.26 Description Phase - RETRIEVE Request Result

```

-----1-----2-----3-----4-----5-----6-----
                MODIFY    SALES

```

```

#                <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>

```

```

-----1-----2-----3-----4-----5-----6-----
-->: 4                DISPLAY FROM LINE: 01
                NAMED FIELD (NF)                LINE: 00                NF-RANK: 00
1:NF   4:RTV                NAME: QUANTITY
2:UF   5:TEST                ATTR: DI    BCYA                COMP:
3:AR   6:MAP    SPIC: 9(3)                INITIAL:
7:QUIT

```



6.1.27 Description Phase - New Location

```

                                MODIFY   SALES
-----1-----2-----3-----4-----5-----6-----
#                               <Product type> #####
<Name of representative:> ##### <Code:> #####
<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>

```

```

-----1-----2-----3-----4-----5-----6-----
-->: 1                                DISPLAY FROM LINE: 01
      NAMED FIELD (NF)                LINE: 12                NF-RANK: 03
1:NF   4:RTV                          NAME: QUANTITY
2:UF   5:TEST                          ATTR: DI           BCYA           COMP:
3:AR   6:MAP   SPIC: 9(3)              INITIAL:
      7:QUIT

```

A new location of the field is made.



6.1.28 Description Phase - RETRIEVE Request

```

                                MODIFY   SALES
-----1-----2-----3-----4-----5-----6-----

#           <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ###  <.>  ##### <.>  ##### <.>

-----1-----2-----3-----4-----5-----6-----
-->: 4           NAMED FIELD (NF)           LINE: 01           DISPLAY FROM LINE: 01
1:NF   4:RTV
2:UF   5:TEST           NAME: product-code           NF-RANK: 01
3:AR   6:MAP   SPIC:           ATTR:           COMP:
7:QUIT           INITIAL:

```

A RETRIEVE of the PRODUCT-CODE field parameters is requested.



6.1.29 Description Phase - RETRIEVE Request Result

-----1-----2-----3-----4-----5-----6-----
 MODIFY SALES

```
#          <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>
```

```
-----1-----2-----3-----4-----5-----6-----
-->: 4          NAMED FIELD (NF)          LINE: 00          DISPLAY FROM LINE: 01
          NF-RANK: 00
1:NF   4:RTV          NAME: PRODUCT-CODE
2:UF   5:TEST        ATTR: DI          BYEL          COMP:
3:AR   6:MAP   SPIC: X(4)          INITIAL:
7:QUIT
```



6.1.30 Description Phase - New Location

```

                                MODIFY   SALES
-----1-----2-----3-----4-----5-----6-----

#                               <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>

-----1-----2-----3-----4-----5-----6-----
-->: 1                                DISPLAY FROM LINE: 01
      NAMED FIELD (NF)                LINE: 12                NF-RANK: 02
1:NF   4:RTV                          NAME: PRODUCT-CODE
2:UF   5:TEST                          ATTR: DI             BYEL             COMP:
3:AR   6:MAP   SPIC: X(4)              INITIAL:
      7:QUIT

```

A new location of the field is entered.



6.1.31 Description Phase - TEST Request

```

                                MODIFY   SALES
-----1-----2-----3-----4-----5-----6-----
#           <Product type> #####

<Name of representative:> ##### <Code:> #####

<.....>
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>    ##### <.>    ### <.>    ##### <.>    ##### <.>

-----1-----2-----3-----4-----5-----6-----
-->: 5           NAMED FIELD (NF)           LINE: 12           DISPLAY FROM LINE: 01
           NF-RANK: 02
1:NF   4:RTV           NAME:
2:UF   5:TEST         ATTR:
3:AR   6:MAP   SPIC:           INITIAL:
7:QUIT

```

A TEST request is asked for.



6.1.32 Description Phase - TEST Request Result

Product type

Name of representative:

code:

```

.....
. Product . Article . Quantity . Price . Total .
.         .         .         .         .         .
.....
.         .         .         .         .         .
.         .         .         .         .         .
.         .         .         .         .         .
.         .         .         .         .         .
.....

```

```

.....
.                                GRAND TOTAL :                                .
.....

```

A QUIT request is now asked for.



6.1.33 End of Description Phase

```

                                MODIFY   SALES
-----1-----2-----3-----4-----5-----6-----
<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total> <.>
<.....>
<.>      <.>      <.>      <.>      <.>      <.>      <.>
<.>##### <.>   ##### <.>   ###  <.>   ##### <.>   ##### <.>

<.....>

                                <.....<.>
                                <.> < GRAND TOTAL : > ##### <.>
                                <.....>

-----1-----2-----3-----4-----5-----6-----
-->: 7          NAMED FIELD (NF)          LINE: 12          DISPLAY FROM LINE: 01
                4:RTV                    NAME:            NF-RANK: 02
1:NF           5:TEST                    ATTR:
2:UF           6:MAP   SPIC:              INITIAL:
3:AR           COMP:
7:QUIT

```

A QUIT request is now asked for.



6.1.34 Save Prompt

```

                                MODIFY   SALES
-----1-----2-----3-----4-----5-----6-----

<.><Product<.> <Article><.><Quantity<.>< Price> <.> <Total>  <.>
<.....>
<.>          <.>          <.>          <.>          <.>          <.>
<.>##### <.>  ##### <.>  ### <.>  ##### <.>  ##### <.>

<.....>

                                <.....>
                                <.> < GRAND TOTAL : > ##### <.>
                                <.....>

-----1-----2-----3-----4-----5-----6-----

Do you want to save ?          1
(1:Yes   0:No)
(2:Return to last request)

```





A. SDPI Interface

A.1 Form Generation

Forms are generated by the MAINTAIN_FORM (MNFORM) utility.

In EDIT mode, the form is described routine by routine in the SDPI form definition language. SDPI is short for Standard Device Programmatic Interface.

In FORMS mode, the form is described pictorially using the full screen terminals which support this mode, in conjunction with the form definition language.

For each form, MAINTAIN_FORM generates these members:

- A reference source member in a binary library (BINLIB). MAINTAIN FORM names this member, formname_SF.
- A terminal-independent object member (formname_of_ANY) or as many terminal dependent object members as there are terminal types for which the form has been loaded (formname of terminaltype). These members are created in a binary library (BINLIB).
- Language members in a source library (SLLIB). These members, which contain data structures, are retrieved by the application using COPY routines or an equivalent, and include members for C Language or macro-instructions for GPL. Three types of members are generated:
 - formnameI defines the form identification. It identifies the form and indicates where to mount the form on the screen,
 - formnameV defines the selection vector. It is used to select variable fields in the form,
 - formnameR defines the data record. This data record is an image of the variable fields of the form.



The object forms are searched for in these binary libraries, according to standard search rules: (BLIB, BINLIB1, BINLIB2, BINLIB3).

You can optionally use a UFAS sequential file (OUTFILE) or a diskette (DK) to store terminal object members.

Access to forms is given to the application through the SDPI primitives. These primitives will call a forms runtime package which handles the presentation of forms, using the object form member created by MAINTAIN_FORM.



A.2 Principles of the SDPI Interface

As regards programs, access to forms mode is made through SDPI routines which make reference to language members. These routines provide a basic way to dialogue with the terminal.

Each routine results in an action being taken, except the CDFIDI routine which is purely informative. This action may result either in issuing some kind of message to the terminal (CDGET, CSEND, CDATTR, CDATTL, CDMECH, CDRELS) or in receiving a message (CDRECV, CDPURGE).

For each action of the first kind, the program can specify an enclosure level in the routine, which is used to specify when the message is to be delivered to the terminal, and, if it is to be delivered immediately, whether the program retains the right to send other messages or gives control to the terminal.

The Receive routine (CDRECV) returns an enclosure level which tells the user or his program whether the message has been issued and whose turn it is to reply.

For each routine, a return code indicates whether it was successfully completed.



A.3 Form Activation

Activating a form consists of displaying on the screen the fixed fields, their initial values, and the initial rendition attributes of the variable fields of a form. To activate a form, the program must issue a CDGET routine referencing a form identification structure that identifies the form and its mode of activation. Four activation modes are available: APPEND, OVERLAY, WINDOW, and ERASE.

In APPEND mode the user can specify where to position the form on the screen, relative to forms already mounted. If a new form is activated, all the forms below the one to which the new form is appended are implicitly released and the area occupied by these forms on the screen is cleared. If a form is requested to be mounted at the top of the screen, the whole screen is cleared. Several occurrences of the same form can be mounted simultaneously on the screen. In this case, each occurrence is identified by an occurrence number that is specified in formnameI and formnameV structures.

In OVERLAY mode all the preceding forms are logically released but the screen is not cleared. These forms become frozen and the new form is displayed on the screen without relocation.

The WINDOW mode is a mode in which each form defines a window on the screen. The window consists of the smallest rectangle that may contain the form beginning in line 1 column 1 up to the maximum line and the maximum column.

The program specifies, in the form identification structure, the location of the window by giving the coordinates of its top left corner. All the forms previously on the screen are frozen, as in OVERLAY mode. Then the contents of the window that will contain the form are erased. The form is relocated so as to be placed in the window and it becomes the new active form.

The main characteristic of the WINDOW mode is that when the CDGET references a form which is already displayed in a window, this form becomes the active form again with its fields restored to their contents when the form was the active form. In this case, the form is always displayed at the same location on the screen and the window coordinates that may be specified are ignored. This gives a visibility similar to a stack of sheets of paper that partially overlay each other and where the user may remove a sheet in the middle of the stack to put it on the top of the stack.

Also, the POPUP mechanism may be used to release only the currently active form. In this case, the form window is reset to its underlying contents and the next form in the stack becomes the new active form.



The ERASE mode is similar to the WINDOW mode, with the difference that it first releases all forms and clears the screen.

A form may be activated in WINDOW mode only if the previous form was activated in one of these modes: ERASE, WINDOW. A form cannot be appended to a form activated in WINDOW or ERASE mode.

The object form may be mounted either from the binary libraries of the step or from the terminal diskette. This choice is transparent to the program which always issues the same routine. The forms runtime package either sends the object form or requests the terminal to mount the form from the diskette, depending on the form generation options.

NOTES:

1. The TERM=ANY option is mandatory for a form to be activated in WINDOW or ERASE mode.
2. Activating a form within an IOF startup procedure is not recommended.
3. The number of forms active or frozen is limited to:
8 in TDS
12 in IOF.
This limit applies but is not checked in OVERLAY mode.
4. A form can be activated from a diskette only in APPEND mode relating an empty screen, or in OVERLAY mode.



A.4 Output Actions

Several routines are available that display a message on the terminal:

- CSEND is used to select variable fields of a form. A selection vector is passed to identify the form and to specify the fields that must be modified.
- CDATTR is used to define a rendition attribute of fields of a form. The form and its fields are identified by a selection vector.
- CDATTL is similar to CDATTR except that it sets a list of attributes instead of only one attribute.
- CDMECH is a control function that acts on the terminal, clearing all unprotected fields or setting off the alarm.
- CDRELS releases all the active forms and sets the terminal to line mode. This routine does not clear the screen. If the screen is to be cleared, this routine must be followed by a CDMECH with RESET option.

A.5 Input Actions

When the terminal user enters data on the screen, his program must issue as many CDRECV routines as there are forms for which data is to be entered. It is recommended that each CDRECV be preceded by a CDFIDI routine which returns the identification of the next form to receive data. The program passes a selection vector to the CDRECV routine to identify this form and to select those fields of the form to receive data.

If a function key has been pressed and a function key field has been implicitly or explicitly declared at form generation time, this field, if selected, is filled with the rank of the function key. The mapping of function keys onto terminal-independent ranks is detailed in the *GCOS7-V6 Forms User's Guide*.

If the message contains information relative to another form, the program is notified through an appropriate enclosure level. If the program does not want to receive the pending data, it must issue a CDPURGE routine.



A.6 Traces

The FORMS runtime package provides traces to debug user programs and to serve as a diagnostic tool in case of problems. These traces are activated through the LOG directive under IOF or by a trace print in TDS.

They consist of:

- A trace of each user routine. This trace is issued after each routine is processed; it displays all the parameters of the routine after its execution (the selection vector and the data record contents when used), as well as the routine's return code.
- A trace of all messages received and sent. These messages are designated as INBOUND and OUTBOUND. They are traced as they are processed by FORMS. Therefore, the traces show the anticipation of input message handling.
- Data structures. These are the Data Map, which contains internal mapping information generated by MAINTAIN_FORM, and the terminal independent Preform structure for dynamic forms.

The last two traces are activated by using the option LOG MODIFY ON=*F or **, or LOG START FOR = SYSTEM.



A.7 SDPI COBOL Statements

This section describes the FORMS mode routines accessible via COBOL statements. For the C Language interface see the manual *C Language System Primitives*, and for the GPL interface see *GPL System Primitives*.

Some parameters are passed to the input or output CD structure.

Since the CD cannot be directly used in a COBOL CALL routine, it must be redefined in the Communication section of an I/O structure.

Such a redefinition is subsequently designated as a "CD" alias.

A.7.1 Data Structures

The FORMS services accept several data structures as parameters. The form identification, the data record and the selection vector are generated by MAINTAIN_FORM and retrieved by COPY in the program.

The input and output Communication descriptions are standard COBOL communication structures. The actual use of these structures by FORMS is detailed below.

The input and output Communication Description (CD) entries must be redefined in the Communication Section, because the CD names cannot be used directly in the CALL verb's parameter list. The redefinitions are:

```
CD cd-name-1 [FOR] INPUT.
01 cd-alias-1.
   02 symbolic-queue-data-name    PIC X(12).    (c)
   02 FILLER                      PIC X(36).    (a)
   02 date-data-name              PIC X(6).     (e)
   02 time-data-name              PIC X(8).     (e)
   02 symbolic-source-data-name   PIC X(12).    (b)
   02 text-length-data-name       PIC 9(4).     (b)
   02 end-key-data-name           PIC X.        (f)
   02 status-key-data-name        PIC XX.       (g)
```

and:

```
CD cd-name-2 [FOR] OUTPUT.
01 cd-alias-2.
   02 destination-count-data-name  PIC 9(4) VALUE 1. (h)
   02 text-length-data-name        PIC 9(4).         (b)
   02 status-key-data-name         PIC XX.           (g)
   02 error-key-data-name          PIC X.            (b)
   02 symbolic-destination-data-name PIC X(12).      (d)
```




NOTES:

- a) The FILLERs in the description are not used.
- b) These fields are not used under IOF.
- c) This field must contain the string "CONSOLE" when calling any FORMS service using the input CD.
- d) This field must contain the string "CONSOLE" when calling any FORMS service using the output CD.
- e) After a CDRECV service, these fields contain the date and the time of day.
- f) After a CDRECV service, this field returns the enclosure level associated with the received message. It may be either of the following:
 - "1" further data, coming from another form, is still to be incorporated in the message
 - "3" all data has been received. Control is given to the program.
- g) After any FORMS service, this field contains the status on completion of the statement. The various status keys are detailed in the following routine descriptions.
- h) This field must be set to 1.



A.7.2 CSEND (Forms Send)

Purpose:

This routine causes data in a data record to be transferred to the endpoint indicated by the selection vector (that is, a logical terminal as seen by the program).

Syntax:

```
CALL "CSEND" USING output-CD-alias,  
                 data-record, enclosure-level, selection vector.
```

Parameters:

output-CD-alias	Refer to the CD description.
data record	The data record associated with the form. This structure is output by MAINTAIN_FORM (formnameR).
enclosure-level	The enclosure level to be associated with the data. This is a one character data item that may take these values: <ul style="list-style-type: none">– End of record: Data is quarantined; no message is sent to the terminal.– End of quarantine: The data relative to this command and all previously quarantined data are sent to the terminal. The application keeps control.– End of interaction: The data relative to this command and all previously quarantined data are sent to the terminal. Control is given to the terminal.
selection-vector	The selection vector. This structure is output by MAINTAIN_FORM (formnameV). The occurrence number field of the selection vector must identify the form occurrence to which the command applies. The other fields of the selection vector must be loaded as follows: "space": do not move the associated data field from the data record.



"S": move the associated data field from the data record.

"C": clear the contents of the field.

After execution, the contents of the selection vector fields are unchanged , except in case of a selected numeric field with invalid contents where the selection vector field is loaded with "A".

Return Codes:

Normal:

0 DONE

AB ALMOST: an error was found in at least one field

AF DONEIDE: invalid selection vector contents

Abnormal:

9A BREAK: a break occurred

AC ARGERR: unexpected parameter

A0 SNDVIOL: turn error

A7 SNDARERR: the selection vector does not match an active form

A1 SEQERR: endpoint not in FORMS mode

97 OPTERR: invalid enclosure level

AG NOMATCH: the creation date of the form does not match the creation date of program structures

A9 DVIDFBID: device not supported

9H DNSPEC: invalid endpoint name

S1 DAMAGED: system error

S1 WRONGITM: invalid object form contents

S1 WRONGSTA: invalid data map contents



A.7.3 CDRECV (Forms Receive)

Purpose:

This routine causes data to be received in the data record according to the selection vector.

Syntax:

```
CALL "CDRECV" USING input-CD-alias,  
                    data-record, wait-indicator, selection-vector.
```

Parameters:

input-CD-alias	Refer to CD description.
data-record	The data record associated with the form. This structure is output by MAINTAIN_FORM (formnameR).
selection-vector	<p>This structure is output by MAINTAIN FORM (formnameV). The form name and occurrence number fields of the selection vector must match a form and occurrence number for which data is available. For each field in which a value is to be received from the terminal, the corresponding selection vector field must be loaded with "S".</p> <p>Such a selection vector field will be modified by the command as follows:</p> <ul style="list-style-type: none">– valid data has been moved to the corresponding field.– no data available for the field; i.e., either the field is not a transmittable field or its contents are set to null or to spaces and the RECEIVE SPACES option was not selected for the form.– this field contains the cursor (may also be returned for a non-selected field).– this field contains the cursor and valid data has been moved to it (R + C).



- this field has the attribute DT or IT and data are received in the field.
- a sign was entered in an unsigned field, no data transferred.
- least significant digits have been truncated, data transferred.
- significant digits would have been truncated, no data transferred.
- incorrect characters input to the field, no data transferred.

O, T and + may be returned only for numeric or numeric edited fields; i.e., fields with a numeric or numeric edited screen picture or with a computational usage (e.g. COMP.1 fields).

All other selection vector fields must be loaded with spaces. If some data is available for a non selected field, the data is not transferred and the corresponding selection vector field is set to L (=lost), otherwise the selection vector field is unaffected.

wait-indicator

one character to be set to "0" (provision for future use).

**Return Codes:**

Normal:

0 DONE

AF DONEIDE: invalid selection vector contents

AB ALMOST: at least one field in error

A8 SKIPPED: some received fields were not selected; they are lost

AI IGNORED: unexpected receive function key

Abnormal:

9A BREAK: a break occurred

AC ARGERR: unexpected parameter

A3 RCVVIOL: turn error

A7 RECARERR: the selection vector does not match an active form

A1 SEQERR: endpoint not in FORMS mode

AG NOMATCH: the creation date of the form does not match the
creation date of program structures

A9 DVIDFBID: device not supported

9H DNSPEC: invalid endpoint name

S1 DAMAGED: system error

S1 WRONGITM: invalid object form contents

S1 WRONGSTA: invalid data map contents



A.7.4 CDGET (Form Activation)

Purpose:

This routine causes a form to be activated.

Syntax:

```
CALL "CDGET" USING output-CD-alias,  
                  format [,enclosure-level].
```

Parameters:

output-CD-alias	Refer to CD description.
format	A structure that identifies the form to be activated and where it has to be mounted on the screen. This structure is output by MAINTAIN FORM (formnameI) and is used as follows:

If APPEND mode is specified (i.e., the mode field of the format structure is set to A), all forms following the form specified by the old form name and occurrence number fields of the format structure are released and cleared. The new form is appended after the old form. If the old form and occurrence number fields specify a name consisting of spaces and an occurrence of zero (default initialization), the new form will be appended to the top of the screen and all other forms are released and cleared.

If OVERLAY mode is specified (i.e., the mode field of the format structure is set to O), the old form and occurrence number fields must specify a name consisting of spaces and an occurrence number of zero. The forms that were active are frozen; i.e., they remain visible on the screen but all their fields become protected and they are no longer addressable from the program. The new form is activated at the top of the screen. There is no checking of overlapping of the newly activated form with other forms.



If WINDOW mode is specified, that is if the field "form-name-MD" is set to "W", the forms that were active are frozen. If the form to be activated is already displayed on the screen with the same occurrence number, this form is put on top of other forms and made active again with its previous contents. Otherwise the "form-name-SL" and "form-name-SC" fields determine the line and column numbers of the top left corner of the rectangle where the form is to be placed. The contents of this rectangle are cleared and the form is made active.

If ERASE mode is specified, that is if the field "form-name-MD" is set to "E", all the forms are released and the screen is cleared. The "form-name-SL" and "form-name-SC" fields determine the line and column numbers of the top left corner of the rectangle where the form is to be placed.

If a CDGET with OVERLAY is followed by one or more CDGETs with APPEND, the subsequent forms will be appended to the overlaying form.

The formname-LL field of the format structure gives the number of lines allocated to the new form. If formname-LL is zero, the number of lines allocated is equal to the number of lines in the form. If formname-LL is not zero and is greater than or equal to the number of lines in the form, the form is completed with the appropriate number of blank lines. If formname-LL is not zero and is less than the number of lines in the form, the status FUNCNAV is returned. This field is interpreted only for the new format structure, that is, the first FILLER must be set to 2. The maximum number of active or frozen forms is 6.

enclosure-level

The enclosure level to be associated with the data. This is a one character data item that may take these values:

- End of record: Data is quarantined; no message is sent to the terminal.
- End of quarantine: The data relative to this command and all previously quarantined data are sent to the terminal. The application keeps control.



- End of interaction: The data relative to this command and all previously quarantined data are sent to the terminal. Control is given to the terminal.

The default is "1".

Return Codes:

Normal:

0 DONE

Abnormal:

9A BREAK:	a break occurred
9J ALREADY:	form already active (last form activated in WINDOW mode)
AC ARGERR:	unexpected parameter
A0 SNDVIOL:	turn error
A4 RECNUF:	the specified form was not found
AE FUNCNAV:	activation mode not available
97 OPTERR:	invalid enclosure level
92 ENTRYOV:	Maximum number of active or frozen forms exceeded
or MSGOV:	expansion area overflow
or SPACEOV:	no more space available for control structures
A9 DVIDFBID:	device not supported
AG NOMATCH:	the creation date of the form does not match the creation date of program structures
9H DNSPEC:	invalid endpoint name
9F CALLVIOL:	environment is neither TDS nor IOF
S1 DAMAGED:	system error
S1 WRONGITM:	invalid object form contents
S1 WRONGSTA:	invalid data map contents



A.7.5 CDRELS (Release All Forms)

Purpose:

This routine releases all active forms and puts the terminal in line mode. The screen is not cleared. This routine also releases all attribute modifications into the forms management structure. No attribute modification can be removed by another CALL after this routine has been performed.

Syntax:

```
CALL "CDRELS" USING output-CD-alias [,enclosure-level].
```

Parameters:

output-CD-alias	Refer to CD description.
enclosure-level	<p>The enclosure level to be associated with the data. This is a one character data item that may take these values:</p> <ul style="list-style-type: none">– End of record: Data is quarantined; no message is sent to the terminal.– End of quarantine: The data relative to this command and all previously quarantined data are sent to the terminal. The application keeps control.– End of interaction: The data relative to this command and all previously quarantined data are sent to the terminal. Control is given to the terminal. The default is "1".



Return Codes:

Normal:

0 DONE

Abnormal:

9A (BREAK):	a break occurred
AC ARGERR:	unexpected parameter
A0 SNDVIOL:	turn error
A1 SEQERR:	endpoint not in FORMS mode
97 OPTERR:	invalid enclosure level
A9 DVIDFBID:	device not supported
9H DNSPEC:	invalid endpoint name
92 MSGOV:	expansion area overflow
S1 DAMAGED:	system error
S1 WRONGITM:	invalid object form contents
S1 WRONGSTA:	invalid data map contents

NOTE:

After calling CDRELS, the released forms are still displayed. Activation of a new form by CALL CDGET in OVERLAY or WINDOW mode could lead to unpredictable results.



A.7.6 CDPURGE (Purge Input Data)

Purpose:

This routine purges all pending input messages and gives control back to the application.

Syntax:

```
CALL "CDPURGE" USING input-CD-alias.
```

Parameters:

input-CD-alias Refer to CD description.

Return Codes:

Normal:

0 DONE

Abnormal:

9A BREAK:	a break occurred
A1 SEQERR:	endpoint not in FORMS mode
9H DNSPEC:	invalid endpoint name
S1 DAMAGED:	system error
S1 WRONGITM:	invalid object form contents
S1 WRONGSTA:	invalid data map contents



A.7.7 CDATTR (Attribute Selection)

Purpose:

This routine causes an attribute to apply to the fields selected in the selection vector. This routine cannot remove an attribute modification performed before a CALL CDRELS.

Syntax:

```
CALL "CDATTR" USING output-CD-alias, selection-vector,  
attribute-identifier, [enclosure-level].
```

Parameters:

output-CD-alias	Refer to CD description.
selection-vector	<p>The selection vector. This structure is output by MAINTAIN_FORM (formnameV). The form name and occurrence number fields of the selection vector must identify the form occurrence to which the command applies. The other fields of the selection vector must be loaded as follows:</p> <p>"space": do not assign the attributes of the associated field.</p> <ul style="list-style-type: none">– assign the attributes of the associated field.– clear the contents of the field.– both clear the contents of the field and assign the attributes as specified. <p>After execution, the contents of the selection vector fields are unchanged.</p>
attribute identifier	<p>The attribute to be applied to the fields selected. This is a four character alphanumeric data item that may take these values:</p> <p>BI: Blink</p> <p>BD: Bold (i.e., high intensity).</p>



Bxxx: Background color
where xxx may be:

RED: Red
YEL: Yellow
BLU: Blue
GRE: Green
CYA: Cyan
MAG: Magenta
WHI: White
BLA: Black
DFT: Default color

CN: Conceal.

COS: Column Separator.

CP: Cursor position.

DFT: Default rendition (i.e.,

NBI, NHL, NRV, NCOS, NUL, BDFT, FDFT, and
normal intensity).

FT: Faint (i.e., reduced intensity).

Fxxx: Foreground color (xxx may take the same values
as for Bxxx).

HL: Rendition highlighted in a terminal-specific
manner.

INIT: Initial attributes.

NBI: No blink

NCOS: No column separator

NHL: Not highlighted.

NPR: Not protected.

NRV: No reverse video.

NTR: Not transmittable.

NUL: Not underlined.

PR: Protected.

RV: Reverse video.

TR: Transmittable.

UL: Underlined.



Once an attribute is modified, it remains modified until either another CDATTR or CDATTL command specifies a contradictory attribute for the same field or a CDMECH command with INITAT argument is executed.

When a CP attribute is applied to a protected field the cursor cannot be positioned on that field, so it will be positioned either on the first field of the screen to which the CP attribute was given at form generation (if any), or on the first non-protected field of the screen (home field).

enclosure-level

The enclosure level to be associated with the data. This is a one character data item that may take these values:

- End of record: Data is quarantined; no message is sent to the terminal.
- End of quarantine: The data relative to this command and all previously quarantined data are sent to the terminal. The application keeps control.
- End of interaction: The data relative to this command and all previously quarantined data are sent to the terminal. Control is given to the terminal.

The default is "1".

Return Codes:

Normal:

0 DONE

AF DONEIDE: invalid selection vector contents.

Abnormal:

9A BREAK: a break occurred

AC ARGERR: unexpected parameter

A0 SNDVIOL: turn error

A7 SNDARERR: the selection vector does not match an active form



A1 SEQERR:	endpoint not in FORMS mode
97 OPTERR:	invalid enclosure level
A6 OBJUNKN:	unknown attribute
AG NOMATCH:	the creation date of the form does not match the creation date of program structures
A9 DVIDFBID:	device not supported
9H DNSPEC:	invalid endpoint name
or CONFLICT:	the attribute conflicts with the form (e.g., because the selected field starts in column 1)
92 MSGOV:	expansion area overflow
S1 DAMAGED:	system error
S1 WRONGITM:	invalid object form contents
S1 WRONGSTA:	invalid data map contents



A.7.8 CDATTL (Attribute List Selection)

Purpose:

This routine causes a list of attributes to apply to the fields selected in the selection vector. Attributes are evaluated sequentially, hence the effect of one attribute may be cancelled by the following attributes in the list.

Syntax:

```
CALL "CDATTL" USING output-CD-alias, selection vector,  
attribute-identifier, [enclosure-level].
```

Parameters:

output-CD-alias	Refer to CD description.
selection-vector	<p>This structure is output by <code>MAINTAIN_FORM</code> (<code>formnameV</code>). The form name and occurrence number fields of the selection vector must identify the form occurrence to which the command applies. The other fields of the selection vector must be loaded as follows:</p> <p>"space": do not assign the attributes of the associated field.</p> <ul style="list-style-type: none">– assign the attributes of the associated field.– clear the contents of the field.– both clear the contents of the field and assign the attributes as specified. <p>After execution, the contents of the selection vector fields are unchanged.</p>



attribute-identifier	<p>A structure with this description:</p> <pre>01 data-name1 . 02 data-name2 PIC 9(3) VALUE n. 02 data-name3 PIC X(4) OCCURS n.</pre> <p>where data-name2 is the number of attributes to be applied coded in decimal and each element of data-name3 is an attribute identifier. The list of attribute identifiers and their interpretations are detailed in the description of CDATTR command.</p>
enclosure-level	<p>The enclosure level to be associated with the data. This is a 1- character data item that may take these values:</p> <ul style="list-style-type: none"> – End of record: Data is quarantined; no message is sent to the terminal. – End of quarantine: The data relative to this command and all previously quarantined data are sent to the terminal. The application keeps control. – End of interaction: The data relative to this command and all previously quarantined data are sent to the terminal. Control is given to the terminal. <p>The default is "1".</p>

Return Codes:

Normal:

0 DONE

AF DONEIDE invalid selection vector contents

Abnormal:

9A BREAK a break occurred

AC ARGERR unexpected parameter

A0 SNDVIOL turn error

A7 SNDARERR the selection vector does not match an active form



A1 SEQERR	endpoint not in FORMS mode
97 OPTERR	invalid enclosure level
A6 OBJUNKN	unknown attribute
AG NOMATCH	the creation date of the form does not match the creation date of program structures
A9 DVIDFBID	device not supported
or CONFLICT	the attribute conflicts with the form (e.g., because the selected field starts in column 1)
9H DNSPEC	invalid endpoint name
92 MSGOV	expansion area overflow
S1 DAMAGED	system error
S1 WRONGITM:	invalid object form contents
S1 WRONGSTA:	invalid data map contents



A.7.9 CDFIDI (Form Identification)

Purpose:

This routine causes the system to return the name of the form relative to the next message.

Syntax:

```
CALL "CDFIDI" USING input-CD-alias, form-identifier.
```

Parameters:

input-CD-alias	Refer to CD description.
form identifier	The name and occurrence of the form relative to the next message (or portion of message) to be received. The format of this structure is as follows: 01 form-identification. 02 form-name PIC X(8). 02 occurrence-number PIC 9(3).

Return Codes:

Normal:

0 DONE

Abnormal:

A3 RCVVIOL:	turn error
9A BREAK:	a break occurred
A9 DVIDFBID:	device not supported
9H DNSPEC:	invalid endpoint name
S1 DAMAGED:	system error
S1 WRONGITM:	invalid object form contents
S1 WRONGSTA:	invalid data map contents



A.7.10 CDMECH (Control Function Mechanism)

Purpose:

This statement activates a control function mechanism. Note that it cannot remove an attribute modification performed before a CALL CDRELS.

Syntax:

```
CALL "CDMECH" USING output-CD-alias, mechanism-identifier  
[, enclosure-level].
```

Parameters:

output-CD-alias	Refer to CD description.
mechanism-identifier	A mnemonic that represents a control function mechanism that is related to the device. This is a 6-character alphanumeric data item that may take these values: ALARM: activate the audio or visual alarm, if any.
CLEAR:	clear all unprotected fields. PROTECT: turn to protected all named fields of all active forms. This command is effective on the next CDGET. INITAT (alias RESET): clear unprotected fields and reset attributes to their initial value. When the screen is in normal mode, (i.e., after CDRELS LEVEL=1), this command clears the screen. INIT: reset all forms to their initial state. STPRA and STPRV: These mechanisms are valid only if the endpoint is a forms mode printer, otherwise they have no effect. If the STPRV mechanism is set, then for all subsequent CDSEND commands, only the variable fields are printed. If the STPRA mechanism is set, then for all subsequent CDSEND commands, all fields are printed. STPRA is the default mode.



CPON: This mechanism sets up a mode where the cursor position is notified in subsequent CDREV statements, for terminals that support this feature (QUESTAR 200, IBM3278/3279, MINITEL, HDS7). The cursor position is indicated by a character in the corresponding field of SELECTION-VECTOR:

"c" cursor position in field not received

"x" cursor position plus the value received in the field.

CPOFF: This mechanism resets a previous CPON mechanism.

POPUP: When the active form has been activated in window mode, this mechanism causes this form to be released and the previous form in the stack of displayed forms to become active. The window associated with the released form is reset to the underlying contents. When the active form has not been activated in WINDOW mode, or when the stack of displayed forms is reduced to one form, a return code FUNCNAV is sent.

TL80ON/TL80OFF: for selecting / deselecting 80 column mode (Minitel in mixed mode only).

HDS7ON/HDS7OFF: for activating / deactivating HDS7 specific features (function key management).

enclosure-level

The enclosure level to be associated with the data. This is a one character data item that may take these values:

- End of record: Data is quarantined; no message is sent to the terminal.
- End of quarantine: The data relative to this command and all previously quarantined data are sent to the terminal. The application keeps control.
- End of interaction: The data relative to this command and all previously quarantined data are sent to the terminal. Control is given to the terminal.

The default is "1".



Return Codes:

Normal:

0 DONE

Abnormal:

9A BREAK:	a break occurred
AC ARGERR:	unexpected parameter
A0 SNDVIOL:	turn error
97 OPTERR:	invalid enclosure level
A6 OBJUNKN:	unknown mechanism
A9 DVIDFBID:	device not supported
9H DNSPEC:	invalid endpoint name
S1 WRONGITM:	invalid object form contents
S1 WRONGSTA:	invalid data map contents
AE FUNCNAV:	mechanism not available



A.8 A Programming Example

This is an example of a COBOL program using a form under IOF. The form is called SALES and is used for recording the sales of representatives. The program calculates the total sales value for each product and a grand total for the sales of all products. If data is entered incorrectly, the fields of the corresponding line are highlighted in a terminal-dependant way and the data can be re-entered.

The form image of SALES is as follows:

```

                # PRODUCT TYPE #####
NAME OF REPRESENTATIVE ##### CODE #####
#
PRODUCT CODE QUANTITY  PRICE    TOTAL
#####  #####  ###    #####  #####
#####  #####  ###    #####  #####
#####  #####  ###    #####  #####
#####  #####  ###    #####  #####
#####  #####  ###    #####  #####
#####  #####  ###    #####  #####
#####  #####  ###    #####  #####
#####  #####  ###    #####  #####
GRAND TOTAL #####

```




The corresponding form definition language is:

```
NF SLASH LINE IS 01 COL IS 02 SCREEN-PIC IS X(1) UL.
NF SALE-TITLE LINE IS 01 COL IS 37 SCREEN-PIC IS X(10) UL.
NF REP-NAME LINE IS 02 COL IS 29 SCREEN-PIC IS X(27) UL.
NF REP-CODE LINE IS 02 COL IS 64 SCREEN-PIC IS X(6) UL DI.
ARRAY SALE-LINES OCCURS 07.
NF PRODUCT LINE IS 04 COL IS 03 SCREEN-PIC IS X(6).
NF PRODUCT-CODE LINE IS 04 COL IS 18 SCREEN-PIC IS X(4) DI.
NF QUANTITY LINE IS 04 COL IS 35 SCREEN-PIC IS 9(3) DI.
NF PRICE LINE IS 04 COL IS 46 SCREEN-PIC IS ZZ9.99 NU.
NF TOTAL LINE IS 04 COL IS 59 SCREEN-PIC IS ZZZ9.99 PR.
UF LINE IS 04 COL IS 13 VALUE IS ":".
UF LINE IS 04 COL IS 27 VALUE IS ":".
UF LINE IS 04 COL IS 42 VALUE IS ":".
UF LINE IS 04 COL IS 55 VALUE IS ":".
END-ARRAY.
NF GRAND-TOTAL LINE IS 15 COL IS 50 SCREEN-PIC IS *****9.99 PR.
UF LINE IS 01 COL IS 23 VALUE IS "PRODUCT TYPE" RV.
UF LINE IS 02 COL IS 04 VALUE IS "NAME".
UF LINE IS 02 COL IS 09 VALUE IS "OF".
UF LINE IS 02 COL IS 12 VALUE IS "REPRESENTATIVE".
UF LINE IS 02 COL IS 58 VALUE IS "CODE".
UF LINE IS 03 COL IS 03 VALUE IS "PRODUCT".
UF LINE IS 03 COL IS 13 VALUE IS ":".
UF LINE IS 03 COL IS 18 VALUE IS "CODE".
UF LINE IS 03 COL IS 27 VALUE IS ":".
UF LINE IS 03 COL IS 31 VALUE IS "QUANTITY".
UF LINE IS 03 COL IS 42 VALUE IS ":".
UF LINE IS 03 COL IS 46 VALUE IS "PRICE".
UF LINE IS 03 COL IS 55 VALUE IS ":".
UF LINE IS 03 COL IS 60 VALUE IS "TOTAL".
UF LINE IS 15 COL IS 36 VALUE IS "GRAND".
UF LINE IS 15 COL IS 42 VALUE IS "TOTAL".
```



These data structures are generated:

SALESI:

```
*
02 FILLER      PIC X VALUE "3".
02 FILLER      PIC X(8) VALUE "SALES".
02 SALES-NO    PIC 9(3) VALUE ZERO.
02 SALES-MD    PIC X VALUE "A".
02 SALES-OF    PIC X(8) VALUE SPACES.
02 SALES-OO    PIC 9(3) VALUE ZERO.
02 SALES-LL    PIC 9(3) VALUE ZERO.
02 FILLER      PIC X(5) VALUE "87097".
02 FILLER      COMP-1 VALUE 01385.
02 SALES-AF    PIC 9 VALUE 1.
02 SALES-SL    PIC 9(3) VALUE 1.
02 SALES-SC    PIC 9(3) VALUE 1.
```

SALESR:

```
04 SALES-FC    PIC 99.
04 SLASH       PIC X(1).
04 SALE-TITLE  PIC X(10).
04 REP-NAME    PIC X(27).
04 REP-CODE    PIC X(6).
04 SALE-LINES-A .
    05 SALE-LINES OCCURS 7.
        06 PRODUCT      PIC X(6).
        06 PRODUCT-CODE PIC X(4).
        06 QUANTITY     PIC 9(3).
        06 PRICE        PIC 999V99.
        06 TOTAL        PIC 9999V99.
04 GRAND-TOTAL PIC 99999999V99.
```



SALESV:

```
02 SALESV.
03 FILLER      PIC X VALUE "2".
03 FILLER      COMP-1 VALUE 41.
03 FILLER      PIC X(8) VALUE "SALES".
03 SALES-VO    PIC 9(3) VALUE ZERO.
03 SALES-V.
  04 SALES-FC-V    PIC X.
  04 SLASH-V      PIC X.
  04 SALE-TITLE-V  PIC X.
  04 REP-NAME-V   PIC X.
  04 REP-CODE-V   PIC X.
  04 SALE-LINES-AV .
    05 SALE-LINES-V OCCURS 7.
      06 PRODUCT-V      PIC X.
      06 PRODUCT-CODE-V PIC X.
      06 QUANTITY-V     PIC X.
      06 PRICE-V        PIC X.
      06 TOTAL-V        PIC X.
04 GRAND-TOTAL-VPIC X.
```



The program, which is called DEMO, is listed below:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. DEMO.
AUTHOR. A N OTHER.
INSTALLATION. GAMBETTA CENTER CENTER.
DATE-WRITTEN. JANUARY 1986.
DATE-COMPILED. JANUARY 1986.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. LEVEL-64.
OBJECT-COMPUTER. LEVEL-64.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 I PIC 999.
77 J PIC 9.
77 W-I PIC 9.
77 ERROR-INDICATOR PIC 9.
77 W-NBF PIC 9.
77 W-ATTRIB PIC X(4) VALUE "INIT".
01 SALES-SW.
    COPY SALESV REPLACING TRAILING "-V" BY "-W".
01 ATTRIBUTE.
    02 FILLER PIC 999 VALUE 002.
    02 FILLER PIC X(4) VALUE "CP".
    02 FILLER PIC X(4) VALUE "HL".
77 TOTAL-MAN PIC 9(4)V99.
77 TOTAL-NUM PIC 9(7)V99.
77 W-ATTRIBUT PIC X(6).
77 W-LEVEL PIC X.
01 MESS-AREA.
    02 FILLER PIC X(8).
01 SALES1.
    COPY SALES1.
01 SALES-SV.
    COPY SALESV.
01 SALESR.
    COPY SALESR.
COMMUNICATION SECTION.
CD IQ INPUT
    QUEUE ISQ
    MESSAGE DATE IMD
    MESSAGE TIME IMT
    SOURCE ISS
    TEXT LENGTH ITL
    END KEY IEK
    STATUS KEY ISK
    COUNT IMC.
```



```
01      RIQ.
02      RISQ  PIC X(12).
02      RISU  PIC X(36).
02      RIMD  PIC 9(6).
02      RISS  PIC X(12).
02      RITL  PIC 9(4).
02      RIEK  PIC X.
02      RISK  PIC XX.
02      RIMC  PIC 9(6).
CD OQ OUTPUT
      DESTINATION COUNT      ODC
      TEXT LENGTH           OTL
      STATUS KEY            OSK
      ERROR KEY             OEK
      DESTINATION           OSD.
01 ROQ.
      02  RODC  PIC 9(4).
      02  ROTL  PIC 9(4).
      02  ROSK  PIC XX.
      02  ROEK  PIC X.
      02  ROSD.
          03 RNET  PIC X(4).
          03 RTRM  PIC X(4).
          03 RTYP  PIC X(3).
          03 RNUM  PIC X.

PROCEDURE DIVISION.
PROG-STAR.
      MOVE "CONSOLE" TO ISQ OSD.
      MOVE 1 TO ODC.
      MOVE 1 TO OTL.
      MOVE "3" TO W-LEVEL.
      MOVE ALL "S" TO SALES-W.
      MOVE ALL SPACES TO SALESR.
CALL "CDGET" USING ROQ SALESI W-LEVEL.
IF OSK = "A4" GO TO END-PROG.
MAIN-LOOP.
      MOVE ALL SPACES TO SALESR.
      MOVE ALL "S" TO SALES-V.
      CALL "CDRECV" USING RIQ SALESR W-LEVEL SALES-SV.
      IF SLASH-V = "R" AND SLASH = "/"
          GO TO END-PROG.
MOVE ALL SPACES TO      SALE-TITLE-V
                        SLASH-V
                        REP-CODE-V
                        REP-NAME-V.
MOVE 0 TO ERROR-INDICATOR.
MOVE ZERO TO TOTAL-NUM.
PERFORM TOTAL-D THRU TOTAL-F VARYING I FROM 1 BY 1
                        UNTIL I > 7.
```



```
IF ERROR-INDICATOR = 1
  MOVE "1" TO W-LEVEL
  CALL "CDATTR" USING ROQ SALES-SW W-ATTRIB W-LEVEL
  MOVE "3" TO W-LEVEL
  MOVE SPACES TO GRAND-TOTAL-V
  CALL "CDATTL" USING ROQ SALES-SV ATTRIBUTE W-LEVEL
  MOVE SALES-V TO SALES-W
  GO TO MAIN-LOOP
ELSE
  MOVE "1" TO W-LEVEL
  MOVE ALL "S" TO SALES-W
  CALL "CDATTR" USING ROQ SALES-SW W-ATTRIB W-LEVEL
  PERFORM PREP-SEND THRU END-PREP-SEND VARYING J
    FROM 1 BY 1 UNTIL J > 7
  MOVE "S" TO GRAND-TOTAL-V
  MOVE TOTAL-NUM TO GRAND-TOTAL
  MOVE "3" TO W-LEVEL
  CALL "CDSEND" USING ROQ SALESR W-LEVEL SALES-SV
  MOVE "S" TO SLASH-V
  CALL "CDRECV" USING RIQ SALESR W-LEVEL SALES-SV.
  IF SLASH-V = "R" AND SLASH = "/"
    GO TO END-PROG.
  MOVE "1" TO W-LEVEL.
  CALL "CDRELS" USING ROQ W-LEVEL.
  MOVE ALL "S" TO SALES-V.
  MOVE ALL SPACES TO SALESR.
  MOVE "3" TO W-LEVEL.
  CALL "CDGET" USING ROQ SALESI W-LEVEL.
  GO TO MAIN-LOOP.
  END-PROG.
  STOP RUN.
TOTAL-D.
  IF PRODUCT-V (I) NOT = "R" GO TO TOTAL-NR.
  IF PRODUCT-CODE-V (I) = "R"
  AND QUANTITY-V (I) = "R"
  AND PRICE-V (I) = "R"
    MULTIPLY QUANTITY (I) BY PRICE (I)
      GIVING TOTAL-MAN
    MOVE TOTAL-MAN TO TOTAL (I)
    ADD TOTAL-MAN TO TOTAL-NUM
    MOVE ALL SPACES TO SALE-LINES-V (I)
    GO TO TOTAL-F.
TOTAL-NR.
  IF PRODUCT-CODE-V (I) = "S"
  AND QUANTITY-V (I) = "S"
  AND PRICE-V (I) = "S"
    MOVE ALL SPACES TO SALE-LINES-V (I)
    GO TO TOTAL-F.
TOTAL-ERROR.
  MOVE 1 TO ERROR-INDICATOR.
  MOVE ALL "S" TO SALE-LINES-V (I).
```



```
TOTAL-F.  
  EXIT.  
PREP-SEND.  
  IF SALE-LINES (J) = SPACES  
    MOVE ALL SPACES TO SALE-LINES-V (J)  
ELSE  
  MOVE ALL "S" TO SALE-LINES-V (J).  
END-PREP-SEND.  
  EXIT.
```





Index

-

- symbol, SCREEN-PICTURE clause 4-9

+

+ symbol, SCREEN-PICTURE clause 4-9

A

activating forms A-4
alphamosaic character sets 5-21
AN see ANY-CHARACTER attribute: 4-14
ANY-CHARACTER attribute 4-14
ARRAY statement 1-14, 4-13
attribute codes 4-14

B

BACKGROUND-COLOR attribute 4-14
BD see BOLD attribute: 4-14
BINLIB parameter
 DELETE command 3-1
 LIST command 3-4
 MOVE command 3-10
BL see BLINKING attribute: 4-14
BLINKING attribute 4-14
BOLD attribute 4-14
Bxxx see BACKGROUND-COLOR
 attribute: 4-14

C

C

 form identification 1-8
 SDPI interface 1-8
CDATTL routine A-25
CDATTR routine A-21
CDFIDI routine A-28
CDGET routine A-15
CDMECH routine A-29
CDPURGE routine A-20
CDRECV routine A-12
CDRELS routine A-18
CSEND routine A-10
character sets
 DKU7005/7 terminals 5-3
 DKU7107 terminals 5-5
 DKU7211 terminals 5-8
 HDS7 terminals 5-10
 IBM3270 terminals 5-15
 IBM3278/3279 terminals 5-19
 PC7800 terminals 5-14
 VIP7804 terminals 5-10
 VIP8800 terminals 5-10
CHARACTER-SET-AM attribute 4-15
CHARACTER-SET-PS attribute 4-15
CN see CONCEALED attribute: 4-16
COBOL
 form identification 1-5
 SDPI interface 1-5
color
 background 4-14
 foreground 4-16
 IBM3278/3279 terminals 5-18



column modes, MINITEL model 1B 5-26
 COLUMN-SEPARATOR attribute 4-16
 comma symbol, SCREEN-PICTURE
 clause 4-8
 commands
 COMPILE 2-18
 CREATE 2-14
 DECOMPILE 2-22
 DELETE 3-1
 DISPLAY 3-3
 EDIT 2-17
 EXTRACT 2-23
 FSE 2-17
 ICREATE 2-1, 6-1
 LIST 3-4
 LOAD 3-6
 MAINTAIN_FORM 1-19
 MODIFY 2-6, 6-13
 MOVE 3-9
 PRINT 3-11
 QUIT 3-11
 TEST 3-12
 TRANSIT 3-12
 COMPILE command
 parameters 2-19
 purpose 2-18
 syntax 2-18
 CONCEALED attribute 4-16
 COS see COLUMN-SEPARATOR
 attribute: 4-16
 CP see CURSOR-POSITION attribute: 4-16
 CREATE command
 parameters 2-15
 purpose 2-14
 syntax 2-14
 creating forms 2-1
 CSAM see CHARACTER-SET-AM
 attribute: 4-15
 CSPS see CHARACTER-SET-PS
 attribute: 4-15
 CURRENCY SIGN IS option 4-3
 currency symbol, SCREEN-PICTURE
 clause 4-9
 CURSOR-POSITION attribute 4-16

D

data-name clause
 NFIELD statement 1-12, 4-6
 UFIELD statement 4-6
 DECIMAL-POINT IS statement 4-3
 DECOMPILE command
 purpose 2-22
 syntax 2-22
 DEFAULT attribute 4-16
 defining
 attributes 6-4
 source forms 4-1
 DELETE command
 parameters 3-1
 purpose 3-1
 syntax 3-1
 DESC statement 4-3
 DETECTABLE attribute 4-16
 DFT see DEFAULT attribute: 4-16
 DI see DIGIT attribute: 4-16
 DIGIT attribute 4-16
 DISPLAY command
 parameters 3-3
 purpose 3-3
 syntax 3-3
 DK parameter
 DELETE command 3-2
 LIST command 3-5
 MOVE command 3-10
 DKU7005/7 terminals
 attributes 5-2
 character sets 5-3
 diskette storage 5-3
 function keys 5-3
 limitations 5-1
 RECEIVE REAL-MODE 5-2
 switches 5-3
 DKU7107 terminals
 attributes 5-4
 character sets 5-5
 function keys 5-5
 limitations 5-4
 RECEIVE REAL-MODE 5-5



- DKU7211 terminals
 - attributes 5-6
 - character sets 5-8
 - function keys 5-7
 - limitations 5-6
 - RECEIVE REAL-MODE 5-7
- DSN parameter
 - DELETE command 3-2
 - LIST command 3-5
 - MOVE command 3-10
- DT see DETECTABLE attribute: 4-16
- E**
- EDIT command
 - purpose 2-17
 - syntax 2-17
- editing rules 4-9
- END-ARRAY statement 1-14, 4-13
- END-FORM statement 4-4
- ENTER ARRAY request 6-6
- ENTER NF request 6-3, 6-23
- ENTER UF request 6-5
- ENTRY-REQUIRED attribute 4-16
- examples
 - ICREATE command 6-1
 - menus 6-1
 - MODIFY command 6-13
 - TEST command 6-11
- EXTRACT command
 - parameters 2-24
 - purpose 2-23
 - syntax 2-24
- F**
- FAINT attribute 4-16
- FLD clause
 - NFIELD statement 4-6
 - UFIELD statement 4-6
- FORCE parameter
 - COMPILE command 2-20
 - DELETE command 3-2
 - LOAD command 3-8
 - MODIFY command 2-8
- FOREGROUND-COLOR attribute 4-16
- form-name parameter
 - COMPILE command 2-19
 - CREATE command 2-15
 - DELETE command 3-1
 - DISPLAY command 3-3
 - EXTRACT command 2-24
 - ICREATE command 2-2
 - LIST command 3-4
 - LOAD command 3-6
 - MOVE command 3-10
 - PRINT command 3-11
 - TRANSIT command 3-12
- form-name1 parameter
 - MODIFY command 2-7
- forms
 - activating A-4
 - COBOL data record 1-6
 - COBOL selection vector 1-7
 - creating 2-1
 - general description 1-1
 - managing 3-1
 - mode A-1
 - modifying 2-1
- FROM parameter
 - DELETE command 3-1
 - LIST command 3-4
 - MOVE command 3-10
- FSE command
 - purpose 2-17
 - syntax 2-17
- FT see FAINT attribute: 4-16
- FUNCTION clause, UFIELD/NFIELD statement 4-12
- function keys
 - DKU7005/7 terminals 5-3
 - DKU7107 terminals 5-5
 - DKU7211 terminals 5-7
 - HDS7 terminals 5-10
 - IBM3270 terminals 5-15
 - IBM3278/3279 terminals 5-20
 - MINITEL terminals 5-24
 - PC7800 terminals 5-12
 - VIP7804 terminals 5-10
 - VIP8800 terminals 5-10



Fxxx see FOREGROUND-COLOR
attribute: 4-16

H

H200 see HEIGHT-RATIO200
attribute: 4-16

HDS7 terminals
attributes 5-9
character sets 5-10
function keys 5-10
limitations 5-9
line options 5-10
RECEIVE REAL-MODE 5-10

HEIGHT-RATIO200 attribute 4-16

HIGHLIGHT attribute 4-17

HL see HIGHLIGHT attribute: 4-17

I

IBM3270 terminals
attributes 5-15
character sets 5-15
function keys 5-15
limitations 5-15
RECEIVE REAL-MODE 5-15

IBM3278/3279 terminals
attributes 5-17
character sets 5-19
color attributes 5-18
configurations supported 5-16
function keys 5-20
light pens 5-20
limitations 5-16
RECEIVE REAL-MODE 5-20

ICREATE command
examples 6-1
parameters 2-2
purpose 2-1
syntax 2-1

IDBD see INPUT-DEVICE-BD device
attribute: 4-17

IMMEDIATE-TRANSMIT attribute 4-17

INPUT-DEVICE-BD device attribute 4-17

IT see IMMEDIATE-TRANSMIT
attribute: 4-17

J

JL see JUSTIFY-LEFT attribute: 4-17

JR see JUSTIFY-RIGHT attribute: 4-17

JUSTIFY-LEFT attribute 4-17

JUSTIFY-RIGHT attribute 4-17

L

LANG parameter
COMPILE command 2-20
CREATE command 2-16
EXTRACT command 2-24
ICREATE command 2-3
LOAD command 3-7
MODIFY command 2-8

language members 1-5

LENGTH clause
NFIELD statement 1-12, 4-7
UFIELD statement 4-7

light pens, IBM3278/3279 terminals 5-20

limitations
DKU7005/7 terminals 5-1
DKU7107 terminals 5-4
DKU7211 terminals 5-6
HDS7 terminals 5-9
IBM3270 terminals 5-15
IBM3278/3279 terminals 5-16
MINITEL basic model 5-23
MINITEL model 1B 5-26
PC7800 terminals 5-11
VIP7804 terminals 5-9
VIP8800 terminals 5-9

LINE IS clause
NFIELD statement 4-6
UFIELD statement 4-6

line options
HDS7 terminals 5-10
VIP7804 terminals 5-10
VIP8800 terminals 5-10



LIST command
 parameters 3-4
 purpose 3-4
 syntax 3-4

LOAD command
 parameters 3-6
 purpose 3-6
 syntax 3-6

location clause
 NFIELD statement 4-6
 UFIELD statement 4-6

LP see PRINTABLE attribute: 4-18

M

MAINTAIN_FORM utility 1-1

managing forms 3-1

MF see MUST-FILL attribute: 4-17

MINITEL basic model
 alphamosaic character sets 5-21

MINITEL terminals
 attributes 5-24
 basic model 5-21
 function keys 5-24
 limitations basic model 5-23
 limitations model 1B 5-26
 model 1B column modes 5-26
 RECEIVE REAL-MODE 5-24

minus symbol, SCREEN-PICTURE
 clause 4-9

MNFORM see MAINTAIN_FORM: 1-1

MODIFY command
 examples 6-13
 parameters 2-7
 purpose 2-6
 syntax 2-6

modifying forms 2-1

MOVE command
 parameters 3-10
 purpose 3-9
 syntax 3-9

MUST-FILL attribute 4-17

N

NAMED-FIELD-CHARACTER
 statement 4-3

NEW parameter, MODIFY command 2-7

NF see NFIELD statement 4-5

NFC see NAMED-FIELD-CHARACTER
 statement: 4-3

NFIELD statement 1-11, 4-5

nine symbol, SCREEN-PICTURE clause 4-8

NLP see NOT-PRINTABLE attribute: 4-17

NO IMPLICIT FUNCTION-CODE
 option 4-2

NOT-PRINTABLE attribute 4-17

NOT-TRANSMITTABLE attribute 4-17

NPR see UNPROTECTED attribute: 4-18

NTR see NOT-TRANSMITTABLE
 attribute: 4-17

NU see NUMERIC attribute: 4-17

NUMERIC attribute 4-17

O

OUTFILE parameter
 DELETE command 3-2
 LIST command 3-4
 MOVE command 3-10

P

PC7800 terminals
 attributes 5-11
 character sets 5-14
 function keys 5-12
 limitations 5-11
 RECEIVE REAL-NODE 5-11

period symbol, SCREEN-PICTURE
 clause 4-8

PLE character set 3-9, 5-5, 5-8

plus symbol, SCREEN-PICTURE clause 4-9

PLW character set 3-9, 5-5, 5-8, 5-19, 5-26

PR see PROTECTED attribute: 4-18



PRINT command
 parameters 3-11
 purpose 3-11
 syntax 3-11
 PRINTABLE attribute 4-18
 PROTECTED attribute 4-18

Q

QUIT command
 purpose 3-11
 syntax 3-11
 QUIT request 6-10

R

RECEIVE REAL-MODE
 DKU7005/7 terminals 5-2
 DKU7107 terminals 5-5
 DKU7211 terminals 5-7
 HDS7 terminals 5-10
 IBM3270 terminals 5-15
 IBM3278/3279 terminals 5-20
 MINITEL terminals 5-24
 PC7800 terminals 5-11
 VIP7804 terminals 5-10
 VIP8800 terminals 5-10
 RECEIVE SPACES option 4-2
 RECORD parameter
 COMPILE command 2-20
 CREATE command 2-16
 EXTRACT command 2-24
 ICREATE command 2-3
 LOAD command 3-8
 MODIFY command 2-8
 rendition attributes
 DKU7005/7 terminals 5-2
 DKU7107 terminals 5-4
 DKU7211 terminals 5-6
 HDS7 terminals 5-9
 IBM3270 terminals 5-15
 IBM3278/3279 terminals 5-17
 MINITEL terminals 5-24
 PC7800 terminals 5-11
 VIP7804 terminals 5-9

VIP8800 terminals 5-9
 REPLACE parameter
 COMPILE command 2-20
 EXTRACT command 2-24
 LOAD command 3-8
 MOVE command 3-10
 TRANSIT command 3-12
 requests
 ENTER ARRAY 6-6
 ENTER NF 6-3, 6-23
 ENTER UF 6-5
 QUIT 6-10
 RETRIEVE 6-7, 6-8, 6-21, 6-24, 6-27
 TEST 6-9, 6-30
 RETRIEVE request 6-7, 6-8, 6-21,
 6-24, 6-27
 REVERSE-VIDEO attribute 4-18
 routines
 CDATTL A-25
 CDFIDI A-28
 CDGET A-15
 CDMECH A-29
 CDPURGE A-20
 CDRECV A-12
 CDRELS A-18
 CSEND A-10
 routines\;CDATTR A-21
 RQ see ENTRY-REQUIRED attribute: 4-16
 RV see REVERSE-VIDEO attribute: 4-18

S

SCREEN-PICTURE clause
 NFIELD statement 1-12, 4-7
 UFIELD statement 4-7
 SDPI
 C interface 1-8
 COBOL interface 1-5
 COBOL statements A-8
 overview 1-1
 principles A-3
 SLLIB parameter, LIST command 3-4
 source form definition language 4-1
 START-FORM statement 4-4
 storage on diskette, DKU7005/7
 terminals 5-3



SUBSTITUTE ATTRIBUTE option 4-3
switches, DKU7005/7 terminals 5-3

T

TERM parameter
 COMPILE command 2-19
 CREATE command 2-15
 EXTRACT command 2-24
 ICREATE command 2-2
 LOAD command 3-6
 MODIFY command 2-7
terminal dependent features 5-1
TEST command
 purpose 3-12
 syntax 3-12
TEST request 6-9, 6-30
TO parameter, MOVE command 3-10
TR see TRANSMITTABLE attribute: 4-18
TRANSIT command
 purpose 3-12
 syntax 3-12
TRANSMITTABLE attribute 4-18

U

UF see UFIELD statement: 4-5
UFIELD statement 1-14, 4-5
UL see UNDERLINED attribute: 4-18
UNDERLINED attribute 4-18
UNPROTECTED attribute 4-18
USAGE IS clause
 NFIELD statement 1-14, 4-11
 UFIELD statement 4-11

V

VALUE IS clause
 NFIELD statement 4-11
 UFIELD statement 1-15, 4-11
VIP7804 terminals
 attributes 5-9
 character sets 5-10
 function keys 5-10
 limitations 5-9

 line options 5-10
 RECEIVE REAL-MODE 5-10
VIP8800 terminals
 attributes 5-9
 character sets 5-10
 function keys 5-10
 limitations 5-9
 line options 5-10
 RECEIVE REAL-MODE 5-10

W

W200 see WIDTH-RATIO200
 attribute: 4-18
WIDTH-RATIO200 attribute 4-18

X

X symbol, SCREEN-PICTURE clause 4-8

Z

Z symbol, SCREEN-PICTURE clause 4-8



Technical publication remarks form

Title :	DPS7000/XTA NOVASCALE 7000 FORMS User's Guide Program Preparation
----------------	-------------------------------------------------------------------

Reference N° :	47 A2 15UJ 05
-----------------------	---------------

Date:	March 2000
--------------	------------

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be promptly investigated by qualified technical personnel and action will be taken as required.
If you require a written reply, please include your complete mailing address below.

NAME : _____ Date : _____

COMPANY : _____

ADDRESS : _____

Please give this technical publication remarks form to your BULL representative or mail to:

Bull - Documentation Dept.
1 Rue de Provence
BP 208
38432 ECHIROLLES CEDEX
FRANCE
info@frec.bull.fr

Technical publications ordering form

To order additional publications, please fill in a copy of this form and send it via mail to:

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

Phone: +33 (0) 2 41 73 72 66
FAX: +33 (0) 2 41 73 70 66
E-Mail: srv.Duplicopy@bull.net

CEDOC Reference #	Designation	Qty
-- -- []		
-- -- []		
-- -- []		
-- -- []		
-- -- []		
-- -- []		
-- -- []		
-- -- []		
-- -- []		
-- -- []		
-- -- []		
[] : The latest revision will be provided if no revision number is given.		

NAME: _____ Date: _____

COMPANY: _____

ADDRESS: _____

PHONE: _____ FAX: _____

E-MAIL: _____

For Bull Subsidiaries:

Identification: _____

For Bull Affiliated Customers:

Customer Code: _____

For Bull Internal Customers:

Budgetary Section: _____

For Others: Please ask your Bull representative.

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

REFERENCE
47 A2 15UJ 05