



# NATURAL

---

## Natural

User's Guide

Version 5.1.1 for UNIX and OpenVMS



This document applies to Natural Version 5.1.1 for UNIX/OpenVMS and to all subsequent releases. Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

© June 2002, Software AG  
All rights reserved

Software AG and/or all Software AG products are either trademarks or registered trademarks of Software AG. Other products and company names mentioned herein may be the trademarks of their respective owners.

# Table of Contents

<b>User's Guide - Overview</b> . . . . .	1
User's Guide - Overview . . . . .	1
Other Natural Documentation . . . . .	1
<b>General Information</b> . . . . .	2
General Information . . . . .	2
Components of Natural . . . . .	3
Invoking Natural . . . . .	4
Terminating Natural . . . . .	5
Natural Main Menu . . . . .	5
Programming Modes . . . . .	6
Overview of Functions . . . . .	6
Help on Error Messages . . . . .	9
Natural Editors - Overview . . . . .	10
Asterisk Notation . . . . .	11
<b>Tutorial - Getting Started With Natural</b> . . . . .	12
Tutorial - Getting Started With Natural . . . . .	12
Selecting a Function or Item . . . . .	13
Session 1 - Creating a Program and a Map . . . . .	14
Session 2 - Creating a Local Data Area . . . . .	27
Session 3 - Creating a Global Data Area . . . . .	34
Session 4 - Creating an External Subroutine . . . . .	39
Session 5 - Editing a Map . . . . .	42
Session 6 - Invoking a Subprogram . . . . .	53
<b>DDM Services</b> . . . . .	59
DDM Services . . . . .	59
DDMs (Data Definition Modules) . . . . .	60
Invoking DDM Services . . . . .	61
Accessing Libraries . . . . .	62
Creating DDMs . . . . .	63
Maintaining DDMs . . . . .	71
Copying DDMs . . . . .	71
Deleting DDMs . . . . .	71
Editing DDMs . . . . .	72
Listing DDMs . . . . .	78
Renaming DDMs . . . . .	79
Services Profile . . . . .	80
Function Keys . . . . .	80
Other Definitions . . . . .	80
<b>Program Editor</b> . . . . .	81
Program Editor . . . . .	81
Editor Basics . . . . .	82
Invoking the Editor . . . . .	82
Leaving the Editor . . . . .	82
Editor Screen . . . . .	82
Editor Profile . . . . .	82
Editor Profile Commands . . . . .	84
Modifying Profile Settings . . . . .	85
Editor Buffer-Pool Settings . . . . .	88
Using Cut and Paste . . . . .	88
Commands . . . . .	90
Editor Commands . . . . .	91
Line Commands . . . . .	120

<b>Data Area Editor</b>	125
Data Area Editor	125
Invoking the Data Area Editor	126
Editor Modes	127
Editing Screen	128
Command Mode	129
Editor Commands	130
Edit Mode	132
Field Types	133
Line Commands	134
<b>Map Editor</b>	136
Map Editor	136
Summary of the Map Creation Process	137
Map Fields	138
Invoking the Map Editor	139
Creating a New Map	139
Editing an Existing Map	139
Map Editor Menu	141
Create	141
Modify	141
Erase	141
Drag	142
Info ON/OFF	142
Lines	142
Ops. Map	142
Quit	143
Creating a Text Constant	144
Creating a User-Defined Variable	145
Using Natural System Variables in a Map	146
Modifying a User-Defined Variable - Field Editing	147
Rule Editing - Processing Rules	149
Array Editing	153
AD - Attribute Definition	156
Selecting Fields from a DDM or User View	158
Defining Fields for a Parameter or Local Data Definition	160
Parameter Definitions	160
Local Data Definitions	160
Map Profile	162
Map Profile Settings	162
Filler Characters	164
Post Assignment	165
Field-Sensitive Processing	166
Advantages of Field-Sensitive Processing	166
Defining a Map as Field-Sensitive	167
<b>SYSMAIN Utility</b>	169
SYSMAIN Utility	169
Libraries	170
Maintaining Libraries	172
List Function	172
Find Function	175
Copy, Move and Rename Functions	176
Delete Function	180
Import Function	181
Invoking SYSMAIN by a Subprogram	184
Direct Commands	184
Additional Keywords for Direct Commands	187

# User's Guide - Overview

This documentation provides information on the usage of Natural in a UNIX or OpenVMS environment.

The following topics are covered:

- **General Information** This section provides information on the following topics: the main components of Natural, how to invoke and how to terminate a Natural session, an overview of the Natural Main Menu functions, an overview of the Natural editors, and how asterisk notation is used.
- **Tutorial - Getting Started with Natural** This section contains a series of tutorial sessions which introduce you to some of the basics of programming in Natural
- **DDM Services** DDMs are data definition modules and represent a logical view of a physical database file. This section describes how to maintain Natural DDMs.
- **Program Editor** This section describes the program editor, which is used to create and maintain Natural programs, subprograms, subroutines, help routines, copycode and text.
- **Data Area Editor** This section describes the data area editor, which is used to create and maintain Natural local data areas, global data areas and parameter data areas.
- **Map Editor** This section describes the map editor, which is used to create and maintain Natural maps and help maps.
- **SYSMAIN Utility** The SYSMAIN utility is used to perform object operations in Natural such as copy, move, delete or import. In most cases this can be accomplished using drag and drop or cut, copy and paste objects. Also importing of objects can be done with this technique.

## Other Natural Documentation

This is a brief overview other Natural documentation available for UNIX and OpenVMS.

### Note:

Documentation for Natural Selectable Units is not included in this overview.

In addition to the User's Guide for UNIX and OpenVMS, the following documentation is available for further information:

- Programming Guide (\*)
- Statements Documentation (\*)
- Reference Documentation (\*)
- Installing and Setting Up Natural on UNIX
- Installing and Setting Up Natural on OpenVMS
- Operations for OpenVMS and UNIX
- Natural Debugger

(\*) These documents apply to all platforms on which Natural can be used.

# General Information

- Components of Natural
- Invoking Natural
- Terminating Natural
- Natural Main Menu
  - Programming Modes
  - Overview of Functions
- Help on Error Messages
- Natural Editors - Overview
- Asterisk Notation

## Components of Natural

Natural provides a complete environment for application development, offering all the functions you need to create and maintain an application:

- the Natural programming language;
- editors to create and maintain programs, maps, data areas and the other types of programming objects that make up a Natural application;
- a utility for maintaining Natural objects;
- a utility to create and maintain error messages to be issued by an application;
- several other utilities for various purposes which you will find helpful when developing an application with Natural.

## Invoking Natural

The way you invoke Natural depends on how the system has been configured at your site. For most installations, you invoke Natural as follows:

- **Under OpenVMS:** Enter the command "NAT51" at your DCL prompt.
- **Under UNIX:** Enter the command "natural" at the UNIX system prompt:  
\$ **natural**

Natural dynamic parameters may be specified with the "NAT51" or "natural" command. For information on dynamic parameters, see *Installing and Setting Up Natural on UNIX* or *Installing and Setting Up Natural on OpenVMS*.

If Natural Security is installed, access to some libraries and some functions may be restricted. Ask your Natural Security administrator for details.

## Terminating Natural

A Natural session can be terminated by any of the following:

- selecting "Fin" on the Main Menu and pressing ENTER,
- entering the system command FIN in the "Direct Commands" window,
- executing a Natural program which contains a TERMINATE statement.

## Natural Main Menu

When you invoke Natural, the Natural Main Menu is displayed:

```

2001-10-25                NATURAL                Library: SYSTEM
12:19:21                  V 5.1.1 Software AG 2001      Mode  : REPORT
User: SAG                  Work Area : empty
.....
•Library      Direct      Services      OS      Fin      •
.....

Select Library
    
```

The header portion of the Main Menu contains the following information.

The top left-hand corner displays:

- the current *date* and *time*;
- the current *user ID*. By default, this is your OpenVMS or UNIX user ID.

The top right-hand corner displays:

- the name of the current library. For more information on libraries, see the section SYSMAIN.
- the current programming mode (reporting mode or structured mode). See the section Programming Modes below for further information.
- the name of the programming object currently in the editor *work area*. This work area is where Natural places a programming object which is to be edited; "empty" indicates that there is no object in the work area.

## Programming Modes

Natural supports two programming modes:

- *Structured mode* is intended for complex applications with a clear and well-defined program structure.
- *Reporting mode* is useful only for the creation of ad hoc reports and small programs which do not involve complex data and/or programming constructs. This mode is not recommended for structured applications or for those applications which could eventually require structuring.

All explanations and examples in this documentation use structured mode. For more information on the differences between reporting and structured mode, see the appendix of the Natural Programming Guide.

The default is reporting mode. To change the mode, select the function "Direct"; a window will appear. In this window, you enter the command "GLOBALS SM=ON" for structured mode, or "GLOBALS SM=OFF" for reporting mode.

## Overview of Functions

The following table provides an overview of the functions available from the Main Menu:

Function	Purpose
<b>Library</b>	Log on to another library or create a new library.
<b>Direct</b>	Issue system commands.
<b>Services</b>	Create DDMs or maintain libraries and the objects they contain.
<b>OS</b>	Start an OpenVMS, UNIX or other command process.
<b>Fin</b>	Terminate the Natural session.

The individual functions are described below.

### Library

The function "Library" is used to create a new Natural library or to log on to an existing Natural library.

See the section SYSMAIN for details.

### Direct

This function is used to execute a Natural system command or a Natural program.

Natural system commands are used to, for example:

- find, access, manipulate, check, and compile objects in a Natural library;
- start, end, and display information about a Natural session;
- access Natural utilities.

If you select this function, a window is displayed in which you enter the name of the system command or program you wish to be executed (provided the program is in your current library, the steplib or the library SYSTEM).

You can also enter a system command in response to a "MORE" prompt. In this case, the program that is being executed will be stopped and the system command will be executed.

**Note:**

Do not confuse Natural system commands, which are used to perform session functions, with Natural statements, which are the components of Natural programs.

The section System Commands contains a detailed description of each system command. The following tables provide an overview of commonly used system commands.

**System Commands to Create and Modify Source Code**

System Command	Purpose
<b>EDIT</b>	Edit the source form of an object.
<b>CLEAR</b>	Clear the contents of the work area of the current editor. The source code currently in the work area is not saved.
<b>CHECK</b>	Check the source code of an object for syntax errors. Syntax checking is also performed as part of the RUN and STOW commands.

**System Commands to Store and Delete Objects**

System Command	Purpose
<b>SAVE</b>	Save the <i>source form</i> of the Natural object currently in the work area of the editor and store it. Syntax is not checked. A saved program can be RUN, but not EXECUTED (see below).
<b>STOW</b>	Save the <i>source form</i> of an object, compile the object and store the resulting <i>object module</i> as well as the source. The object is syntax checked during the compilation process.
<b>SCRATCH</b>	Delete the <i>source and object form</i> of an object. A list of all objects stored in the current library will be displayed; on the list you may then mark the object(s) to be deleted.

**System Commands to Execute Programs**

System Command	Purpose
<b>RUN</b>	Compile and execute a source program, but not a program stored in object form.
<b>EXECUTE</b>	Execute a program that has been compiled and stored in object form.

**Services**

If you select the function "Services", the following screen is displayed:

2001-10-25	NATURAL	Library: SYSTEM
12:22:09	V 5.1.1. Software AG 2001	Mode : REPORT
User: SAG		Work Area : empty
.....		
•Library	Direct	<b>Services</b> OS Fin •
.....		
.....		
•D DDM Services •		
•S SYSMAIN •		
.....		

You can select one of the following functions:

Function	Purpose
<b>DDM Services</b>	Data Definition Modules (DDMs) can be created/edited, and various other DDM maintenance functions can be performed. These functions are described in the section DDM Services.
<b>SYSMAIN</b>	Various library-related functions can be performed. These functions are described in the section SYSMAIN.

## OS Prompt

### Under OpenVMS

Under OpenVMS, the OS prompt function is used to create an OpenVMS subprocess.

If you select this function, the DCL prompt for the subprocess appears. You can then enter an OpenVMS system command or program name.

To return to the current Natural session, enter the command "logout" at the DCL prompt.

#### Note:

If you should not be able to create a subprocess, this is due to the NATPARM option "Start SHELL on OS from Natural" being set to "N"; in this case, contact your Natural administrator.

### Under UNIX

Under UNIX, the OS prompt function is used to start another UNIX command process.

If you select this function, the UNIX system screen appears. You can then enter any UNIX system command or program name.

To return to the current Natural session, enter the command "exit" at the operating-system prompt.

#### Note:

If you should not be able to start another UNIX command process, this is due to the NATPARM option "Start SHELL on OS from Natural" being set to "N"; in this case, contact your Natural administrator.

## Fin

The "Fin" function is used to terminate the Natural session.

## Help on Error Messages

To get more information on an error message displayed by Natural, issue the following system command:

**HELP *nnnn***

where *nnnn* is the number of the error message.

An extended message text will be displayed, which provides more detailed information about the error. For more information see system command HELP.

## Natural Editors - Overview

Natural provides four editors: the *program editor*, the *data area editor*, the *map editor* and the *DDM editor*. The editors are invoked with the system command EDIT. The editor invoked depends on the type of object you specify. If you specify an object by name, the appropriate editor is automatically invoked.

<b>Editor</b>	<b>Creates and Maintains</b>
<b>Program Editor</b>	Natural programs, classes (NaturalX), subroutines, subprograms, help routines, copycode, and text.
<b>Data Area Editor</b>	Global data areas, local data areas and parameter data areas. This editor has a columnar format that is designed for defining the data and object data areas (NaturalX) used in Natural programs or routines.
<b>Map Editor</b>	Maps (screen layouts) referenced in a program's INPUT or WRITE statement. This editor allows direct manipulation of the fields used in an input or output map; moreover, processing rules can be attached to fields in the map.
<b>DDM Editor</b>	DDMs (data definition modules).

The section Tutorial - Getting Started with Natural illustrates the use of the editors. Detailed descriptions of all four editors are provided in the corresponding sections later in this documentation.

When you switch from one editor to another, all changes will be lost unless previously saved. The editing area is overwritten by the new object to be edited.

## Asterisk Notation

Many Natural functions display lists of objects. Usually these lists contain all objects available (for example, all objects of a given type, all objects in a given library, etc.). If you do not wish all objects to be listed, but only a certain range of objects, you may specify that range by using *asterisk notation*.

By specifying a parameter value followed by an asterisk (\*) you will get a list of only those objects whose names (or IDs or whatever the parameter is) begin with that value. This option to enter a value followed by an asterisk is referred to as *asterisk notation*.

### Example 1:

If you enter the system command SCRATCH without any parameters:

**SCRATCH**

you will get a list of all objects in the current library. You can mark those which are to be deleted.

### Example 2:

If you enter the system command SCRATCH as follows:

**SCRATCH BOC\***

you will get a list of only those objects in the current library whose names begin with "BOC".

# Tutorial - Getting Started With Natural

This tutorial is designed to provide a gradual exposure to specific features of the Natural programming environment and illustrate how applications can be modularized. It is *not* intended to provide an example of how an application should be built.

These sessions also represent a general introduction to how the editors may be used. Therefore explanations are kept to a minimum. For a full description of all editor functions and features, please refer to corresponding sections later in this documentation. This tutorial is *not* intended to be a comprehensive description of the full range of possibilities provided by the Natural editors.

The tutorial consists of the following steps:

- Selecting a Function or Item
- Session 1 - Creating a Program and a Map
- Session 2 - Creating a Local Data Area
- Session 3 - Creating a Global Data Area
- Session 4 - Creating an External Subroutine
- Session 5 - Editing a Map
- Session 6 - Invoking a Subprogram

**Note:**

To perform all steps of this tutorial, the database must have been started.

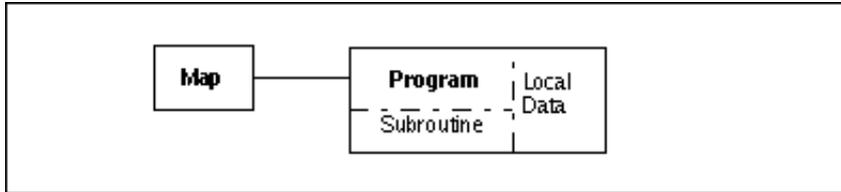
## Selecting a Function or Item

There are two ways to select a function or item from a Natural menu or selection list:

1. You select the function/item name with the arrow keys so that the function/item name is highlighted, and then press the ENTER key.
2. You press the alphabetical key that corresponds to the first character of the function/item name. In this case, you need not press ENTER (key-sensitive selection).  
For example, to select the Create function from the Map Editor Menu, you simply press the key for the letter "C".

So, when this tutorial uses the wording "select the function *xyz*", it is up to you to choose the selection method you prefer. To start with, it may be more convenient to use the first selection method so as to get acquainted with the structure and sequence of menus and selection windows. Once you are familiar with that, it is usually faster to use the second method and press a single alphabetical key (instead of scrolling through a whole list to the desired function and then pressing ENTER).

# Session 1 - Creating a Program and a Map



In this session, you will use the *program editor* to create a Natural *program*. In this first session, the fields used in the program are defined as local data within the program. Moreover, an inline subroutine is contained within the program.

Also, you will use the *map editor* to create a Natural *map* (screen layout). This map will be invoked by the program and displayed on the screen for the user to enter the input data required for the processing of the program.

## Step 1

Invoke Natural according to the procedures at your site. The Natural Main Menu will be displayed:

```

2001-10-25                NATURAL                Library: SYSTEM
12:36:48                  V 5.1.1 Software AG 2001  Mode   : REPORT
User: SAG                  Work Area : empty
.....
·Library      Direct      Services      OS      Fin      ·
.....
  
```

In Natural, you can perform a function either by selecting it from a sequence of menus and selection windows (as shown in Step 3), or by entering a Natural system command in the Direct Command window (as shown in Step 2). The system commands that are available are described in the section System Commands.

## Step 2

Natural offers two modes of programming: *structured mode* and *reporting mode*.

Generally, it is recommended to use structured mode exclusively, because it provides for more clearly structured applications. Therefore all explanations and examples in this section refer to structured mode. Any peculiarities of reporting mode will not be taken into consideration.

The mode currently in effect is indicated on the Natural Main Menu:

```

2001-10-25                NATURAL                Library: SYSTEM
12:39:05                  V 5.1.1 Software AG 2001  Mode   : REPORT
User: SAG                  Work Area : empty
.....
·Library      Direct      Services      OS      Fin      ·
.....
  
```

You must be operating in *structured mode* to work through the sessions in this section.

If the current mode is reporting mode, select "Direct" with the cursor and press ENTER to invoke the Direct Command window:

```

2001-10-25          NATURAL          Library: SYSTEM
 12:43:25          V 5.1.1 Software AG 2001  Mode  : REPORT
User: SAG          Work Area : empty
.....
·Library          Direct          Services          OS          Fin          ·
.....

          ..... Direct Command .....
          ·
          .....
    
```

In this window, enter the following command:

**GLOBALS SM=ON**

and press ENTER; the mode changes to structured mode. Then press ESC.

### Step 3

Natural user-written applications are stored in *libraries*. It may be necessary to move from one library to another in order to perform a maintenance function or work on a different application.

The ID of the current library is shown on Natural Main Menu:

```

2001-10-25          NATURAL          Library: SYSTEM
 12:47:12          V 5.1.1 Software AG 2001  Mode  : STRUCTURED
User: SAG          Work Area : empty
.....
·Library          Direct          Services          OS          Fin          ·
.....
    
```

To move to another library, select "Library" on the Main Menu and press ENTER. A selection window will be displayed, listing all libraries:

```

2001-10-25          NATURAL          Library: SYSTEM
 12:50:56          V 5.1.1 Software AG 2001  Mode  : STRUCTURED
User: SAG          Work Area : empty
.....
·Library          Direct          Services          OS          Fin          ·
.....
.....
· <LOGON>          ·
· DEMO             ·
· DEMO2            ·
· ORDERS           ·
· PRE              ·
· SYSEXP           ·
· SYSEXRM          ·
· SYSTEM           ·
.....
Select Library
    
```

With the arrow keys, scroll through the list until the library "SYSEXP" is selected, and press ENTER.

The "Library" field in top right-hand corner now shows the new library ID. Moreover a list of all objects in that library is displayed.

Instead of scrolling through the list of libraries to select the desired library, you have the following alternatives:

- select the item <LOGON> at the top of the library list and press ENTER; a window will then be displayed, in which you enter the ID of the desired library and press ENTER.
- select "Direct" from the Main Menu, and then enter the command "LOGON *library-ID*" (*library-ID* being the ID of the desired library) in the Direct Command window and press ENTER.
- Enter the first letter of a library (fast selection) to scroll through a list of libraries beginning with this letter.

#### Step 4

To create or modify a Natural program, you use the *program editor*.

By default, this is the *Natural* program editor. However, it is also possible to use another editor (this depends on the setting of the profile parameter EDITOR in your Natural parameter file). This tutorial assumes that the Natural program editor is used.

The first item on the list of objects is <DIRECT COMMAND>, and it is already selected with the cursor. Press ENTER to invoke the Direct Command window. Then invoke the program editor by entering the following command in the Direct Command window:

#### **EDIT PGM01 or in short form: E PGM01**

- *If you have access to the program "PGM01", the program editor will be invoked and the program "PGM01" be read into the editing area. Make sure that the program matches the one shown below.*
- *If you do not have access to program "PGM01", you will receive an error message upon entering the above command. In this case, enter the following command in the Direct Command window:*

#### **EDIT PROGRAM or in short form: E P**

This will invoke the program editor with an empty editing area. Type in the program as shown below.

As you fill up the screen, more blank lines will appear automatically, or you can enter "I" in the prefix area of the editor (where there are usually the line numbers) for more blank lines.

When you have typed in the program, enter the command "CHECK" in the command line of the editor to make sure the program contains no errors. Correct any errors that may be indicated. If necessary, enter the CHECK command again until no further errors are indicated.

Then enter the command "SAVE PGM01" to store the source code of the program under the name "PGM01".

#### **Program PGM01:**

```

* Example Program PGM01
* -----
DEFINE DATA
  LOCAL
  01 #NAME-START      (A20)
  01 #NAME-END        (A20)
  01 #MARK            (A1)
  01 PERSON-VIEW VIEW OF EMPLOYEES
    02 PERSONNEL-ID (A8)
    02 NAME          (A20)
    02 DEPT          (A6)
    02 LEAVE-DUE     (N2)
END-DEFINE
*
REPEAT
*
  INPUT USING MAP 'MAP01'
*
  IF #NAME-START = '.'
    ESCAPE BOTTOM
  END-IF
  MOVE #NAME-START TO #NAME-END
*
  RD1. READ PERSON-VIEW BY NAME
    STARTING FROM #NAME-START
    ENDING AT #NAME-END
    IF LEAVE-DUE >= 30
      PERFORM MARK-SPECIAL-EMPLOYEES
    ELSE
      RESET #MARK
    END-IF
    DISPLAY NAME 3X DEPT 3X LEAVE-DUE 3X '>=30' #MARK
END-READ
*
  IF *COUNTER (RD1.) = 0
    REINPUT 'PLEASE TRY ANOTHER NAME'
  END-IF
*
END-REPEAT
*
  DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
    MOVE '*' TO #MARK
  END-SUBROUTINE
END

```

Once you reached this point (either by writing the program yourself or by reading it into the editor), note the following aspects of a Natural program:

- The first statement must always be a DEFINE DATA statement. All variables to be used in the program must be defined in this initial DEFINE DATA statement.
- All statements which initiate a logical construct or processing loop (DEFINE DATA, REPEAT; IF, READ) must be ended with a corresponding END-... statement (END-DEFINE, END-REPEAT, END-IF, END-READ).
- The READ statement is marked with a so-called *statement label*, namely "RD1.". Using this label, it is possible to reference the statement at a later point in the program (see last IF statement).

When this program is executed, a screen (map) is displayed, prompting the user to enter a name. The EMPLOYEES file is searched to locate all employees with that name. Then a report is displayed which includes the name, department and leave due of each employee with that name. Those employees who have 30 or more days leave due are marked with an asterisk.

The prompting screen is invoked using the INPUT USING MAP statement. The output report is formatted according to information in the DISPLAY statement. The processing required to show which employees have 30 or more days leave is handled in the portion of the program starting with "IF LEAVE-DUE...". Those with 30 or more days of leave due have an asterisk in the final report as a result of processing in the PERFORM statement and the DEFINE SUBROUTINE statement.

**Step 5**

The program contains an INPUT USING MAP statement which invokes a map named "MAP01". This map is yet to be created.

In the command line of the program editor, enter the following command:

**EDIT MAP or in short form: E M**

and press ENTER. This will invoke the Natural map editor, and the Map Editor Menu will be displayed:

```

.....
.                NATURAL MAP EDITOR (Esc to select field)                .
·Create   Modify   Erase   Drag   Info OFF Lines   Ops. Map   Quit   ·
.....

Create a new field definition
    
```

This menu is the main menu of the map editor.

**Step 6**

The first step is to define a *text constant* on the map.

On the Map Editor Menu, select the "Create" function with the arrow keys, and press ENTER.

The Create selection window will be displayed:

```

A Parameter Data Area
G Global Data Area
H Help Routine
L Local Data Area
M Map
N Subprogram
P Program
S Subroutine
T Text Constant
U User Defined
V View Defined
1 Parm Defined
2 Local Defined
    
```

From this window, you select the type of field you wish to create. For example, it is possible to use fields defined in data areas, help routines, programs, subprograms, subroutines, views (DDMs) and other maps, and include them as fields in the map you are editing.

With the arrow keys, select "Text Constant" from the selection window and press ENTER.

With the cursor position, you determine where you wish a field to be placed on the map. By default, the cursor is placed in the top left-hand corner of the map editing screen.

At this position, enter the following text:

```

PERSONNEL INFORMATION
    
```

When you have finished typing the text, press ENTER.

Press PF2 to display an Attribute/Colour Definition window.

Use the DOWN ARROW or DOWN ARROW keys to scroll through the available display attributes and select "Default" (this will cause the field to be displayed "normally", that is, neither intensified nor in any way highlighted); then press ENTER twice to redisplay the Map Editor Menu.

### Step 7

The next step is to add two *Natural system variables* to the map.

On the Map Editor Menu, select the Create function with the arrow keys and press ENTER.

From the Create selection window, select "User Defined" with the arrow keys and press ENTER.

The Extended Field Editing window will be displayed:

```

Extended Field Editing
Field :
Format: A Len:          AL:          PM:          ZP: N  SG: N
Rules : 0 Rule Editing: N Array:      Array Editing: N Mode:
AD:          CD:          CV:          DY: N  HE: N
EM:
    
```

At the same time, a message will prompt you to:

```

Position the cursor and press Enter or format char.(Esc=Cancel)
    
```

It is not possible for two fields to occupy the same position on a map. Therefore position the cursor to an empty location on the map - in this case to the beginning of the line below the text you entered in the previous step - and press ENTER.

A line is displayed indicating the maximum length available for a field, and at the same time a selection window is displayed listing the possible field formats:

A	Alphanumeric
B	Binary
D	Date
F	Floating Point
I	Integer
L	Logical
N	Numeric
P	Packed Numeric
T	Time
*	System

With the arrow keys, select "System" from the window, and press ENTER. A list of all Natural system variables will be displayed.

**Example:**

*APPLIC-ID	A8
*APPLIC-NAME	A32
*COM	A128
*CONVID	I4
*CPU-TIME	I4
*CURSOR	N6
*CURS-COL	P3
*CURS-LINE	P3
*DAT4D	A10
*DAT4E	A10
*DAT4I	A10
*DAT4J	A7
*DAT4U	A10
*DATA	N3
*DATD	A8
*DATE	A8
*DATG	A15
*DATI	A8
*DATJ	A5
*DATN	N8
*DATU	A8
*DATV	A11
*DATVS	A9
*DATX	D
*DEVICE	A8
*ERROR-LINE	N4
*ERROR-NR	N7
*ERROR-TA	A8
*ETID	A8
*HARDCOPY	A8
*HARDWARE	A16
*INIT-ID	A8
*INIT-PROGRAM	A8
*INIT-USER	A8
*LANGUAGE	N1
*LEVEL	N2
*LIBRARY-ID	A8

With the arrow keys, scroll through the list to select the system variable DATX, and press ENTER. When the map is executed, this system variable will display the current date.

The system variable will be included as a field in the map, which now looks as follows:

```
PERSONNEL INFORMATION
DD/DD/DD
```

Press ENTER twice and the definition of the system variable field is complete.

Then press ENTER again to redisplay the Map Editor Menu.

### Step 8

Repeat the previous step, only this time for the system variable \*TIMX. When the map is executed, this system variable will display the current time.

The map should then look as follows:

```
PERSONNEL INFORMATION
DD/DD/DD
TT:TT:TT
```

Press ENTER to redisplay the Map Editor Menu.

### Step 9

Select the Create function. From the Create selection window, select "Text Constant".

In the line below the \*TIMX field (TT:TT:TT), enter the following text:

```
PLEASE ENTER STARTING NAME :
```

Select a display attribute for the entered text in the same manner as you did for the text "PERSONNEL INFORMATION".

The map now looks as follows:

```
PERSONNEL INFORMATION
DD/DD/DD
TT:TT:TT
PLEASE ENTER STARTING NAME :
```

Press ENTER twice to redisplay the Map Editor Menu.

### Step 10

Select the Create function. From the Create selection window, select "User Defined".

The Extended Field Editing window will be displayed:

```
Extended Field Editing
Field :
Format: A Len:           AL:           PM:           ZP: N  SG: N
Rules : 0 Rule Editing: N Array:       Array Editing: N Mode:
AD:           CD:           CV:           DY: N  HE: N
EM:
```

Position the cursor leaving one blank between the text ("PLEASE ENTER STARTING NAME:") and the cursor position and press ENTER.

From the selection window of field formats, select the format "Alphanumeric".

In the Extended Field Editing window, the selected format (A) is now entered and a default name (#1) is assigned to the field.

A message will prompt you to use the cursor keys to size the field to the desired length.

Originally, the length is set to "1" as indicated after the format. You can define the length of the field in the following way:

- Enter the length as a number after the format in the Extended Field Editing window (here, after "AL", Alphanumeric Length).

In this case, enter the length as "20" in the Extended Field Editing window. Press ENTER.

The number of Xs on the map indicates the length of the field.

Now specify a name for the field. Type over the "#1" and enter "#NAME-START" as the field name.

Use the TAB key to select the item "AD=" from the Extended Field Editing window. The Attribute Definition window will be displayed:

```

..Attribute Definition..
·Representation      ·
·Alignment           ·
·I/O Characteristics ·
·Mandatory Characters ·
·Length Characteristics ·
·Upper/Lower Case    ·
·Filler Character    ·
.....

```

Select "I/O Characteristics". The following window will be displayed:

```

A  Input, non-protected
M  Output, modifiable
O  Output, protected

```

Select "M - Output, modifiable". Press ENTER.

The "M" is added to the attributes in the AD field in the Extended Field Editing window. Press ESC when the setting is correct.

Then select "Filler Character" from the Attribute Definition window.

You are then prompted to enter a filler character. Enter an underscore "\_" and press ESC. This will cause any empty positions in the field to be filled with underscores when the map is executed. This enables the user to see the exact position and length of the field, which makes entering input easier.

Press ESC again to finish the attribute definition. The map now looks as follows:

```

PERSONNEL INFORMATION
DD/DD/DD
TT:TT:TT
PLEASE ENTER STARTING NAME: XXXXXXXXXXXXXXXXXXXXX

```

Press ESC to redisplay the Map Editor Menu.

### Step 11

On the Map Editor Menu, select the option "Ops. Map". A selection window with the following functions will be displayed:

```

C Check Map
E Edit Map
K Key Rules
L List Map
P Prof. Map
R Read Map
S Save Map
T Test Map
W Stow Map

```

Select the function "Test Map". Press ENTER.

The map will be tested, and displayed on the screen in the same way as if it were invoked from a program:

```

PERSONNEL INFORMATION
2001-10-25
13:01:22
PLEASE ENTER STARTING NAME: _____

```

With the TAB key, move the cursor to the field after "PLEASE ENTER STARTING NAME:", type in a name and press ENTER.

Keep pressing ENTER until testing is finished and the map editing screen is displayed again.

Then press ENTER to redisplay the Map Editor Menu.

### Step 12

Now that the map is complete, it must be stored in compiled form, so that it can be invoked by a program.

Select "Ops. Map". From the selection window, select the function "Stow Map". The Stow window will be displayed:

```

...Stow Map As: ....
.Name...: .
.Library: SYSEXP .
.....

```

In this window, type in the name "MAP01" and press ENTER to stow the map under that name.

The Stow function stores the map in source and object form; that is, it stores the source code of the map, compiles it, and stores the resulting object module.

Upon completion of the Stow function, the map is displayed in the editor work area.

Press ESC to redisplay the Map Editor Menu.

Then select "Quit" to redisplay the Natural Main Menu.

**Step 13**

On the Main Menu, select "Direct", and then enter the following command in the Direct Command window:

**EDIT PGM01 or in short form: E PGM01**

This command invokes the program editor and reads the program PGM01 into the work area.

In the command line at the top of the program editor, enter the command:

**RUN or in short form: R**

This command compiles and executes the program currently in the work area, PGM01.

A screen will be displayed requesting you to enter a name.

For demonstration purposes, enter the name "MCKENNA". As a result, the message "Please try another name" will be displayed, because the EMPLOYEES file does not contain a record for a person of that name.

Then enter the name "SMITH". Based on this name, the program will produce the following output report:

PAGE	1			2001-10-25	13:10:47
	NAME	DEPARTMENT	LEAVE	>=30	
		CODE	DUE		
	-----	-----	-----	----	
	SMITH	SYSA05	30	*	
	SMITH	MGMT10	12		
	SMITH	SALE40	8		
	SMITH	MGMT10	8		
	SMITH	SALE20	7		
	SMITH	MGMT30	8		
	SMITH	TECH10	4		
	SMITH	TECH10	4		
	SMITH	TECH05	8		
	SMITH	TECH10	8		
	SMITH	TECH10	8		
	SMITH	TECH10	4		
	SMITH	FINA01	30	*	
	SMITH	MGMT01	30	*	
	SMITH	FINA01	28		
	SMITH	SALE02	28		
	SMITH		0		

PAGE	1			2001-10-25 13:14:16
	NAME	DEPARTMENT CODE	LEAVE DUE	>=30
	-----	-----	-----	----
	SMITH	SYSA05	30	*
	SMITH	MGMT10	13	
	SMITH	SALE40	31	*
	SMITH	MGMT10	14	
	SMITH	MGMT10	18	
	SMITH	SALE20	36	*
	SMITH	MGMT30	22	
	SMITH	TECH10	38	*
	SMITH	TECH10	10	
	SMITH	SALE20	14	
	SMITH	TECH05	20	
	SMITH	MGMT10	29	
	SMITH	TECH10	33	*
	SMITH	TECH10	12	
	SMITH	TECH10	19	
	SMITH	FINA01	30	*
	SMITH	MGMT01	30	*

Keep pressing ENTER until you get to the first screen again. When the program asks you again to enter a name, type a period (.) and delete the remaining characters from the input field. When you press ENTER, you will redisplay the program editor.

### Step 14

Whenever you CHECK, RUN, or STOW a program, the program is checked for syntax errors that would keep the program from running.

To introduce such an error, move the cursor to the following statement line:

```
DISPLAY NAME 3X DEPT 3X LEAVE-DUE 3X '>=30' #MARK
```

Delete the apostrophe after "30". The line now looks as follows:

```
DISPLAY NAME 3X DEPT 3X LEAVE-DUE 3X '>=30 #MARK
```

In the command line, enter the command "CHECK". The following error message will be displayed:

```
NAT0305 Text string must begin and end on the same line.
```

Natural requires that a text string (in this case '>=30') must be begun and closed on the same statement line; the beginning and closing of a text string must be indicated by apostrophes. When the closing apostrophe is deleted, this condition is no longer met.

### Step 15

Type in the missing apostrophe again.

Then enter the command "CHECK" again to make sure the program is now correct.

Then enter the command "RUN". When the program has run successfully, display the program again by entering a period ".".

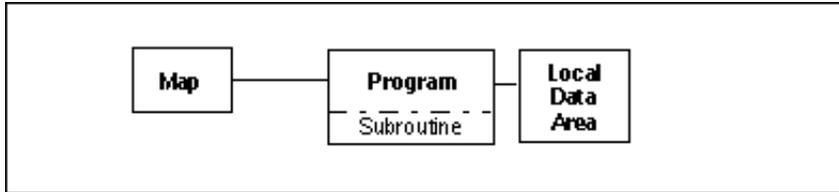
### Step 16

Enter the command "STOW" to store the program in source and (compiled) object form.

Then enter a "." in the command line to redisplay the Natural Main Menu.

End of Session 1.

## Session 2 - Creating a Local Data Area



In Session 1, the fields used by the program were defined within the DEFINE DATA statement in the program itself. However, it is also possible to place the field definitions in a *local data area* outside the program, with the program's DEFINE DATA statement referencing that local data area by name. For a clear application structure, it is usually better to define fields in data areas outside the programs.

In this session, you will relocate the information in the DEFINE DATA statement to a local data area outside the program. In subsequent sessions some of this information can be used as the basis of a global data area shared by a program and an external subroutine. As you will see in Sessions 3 and 4, an important advantage of data areas is to allow a program and its external subroutine to share the same data in a single data area.

### Step 1

Select "Direct" on the Natural Main Menu, and then enter the following command in the Direct Command window:

**EDIT LOCAL or in short form: E L**

In this editor, you define the data areas to be used by Natural programs or routines.

The *data area editor* will be invoked with the object type set to "LOCAL":

```

                                Press <ESC> to enter command mode
Mem: empty      Lib: SYSEXPB   Type: LOCAL      Bytes:      0  Line:      0 of:      0
C T  Comment
*   *** Top of Data Area ***
*   *** End of Data Area ***

F 1 HELP      F 2 CHOICE    F 3 QUIT      F 4 SAVE      F 5 STOW      F 6 CHECK
F 7 READ      F 8 CLEAR      F 9 MEM TYPE  F10 GEN       F11 FLD TYPE  F12
  
```

### Step 2

Enter the command "I" (for Insert) in the first column of the editor:

```

                                Press <ESC> to enter command mode
Mem: empty      Lib: SYSEXPB   Type: LOCAL   Bytes:    0  Line:    0 of:    0
C T   Comment
I *   *** Top of Data Area ***
*   *** End of Data Area ***
    
```

The following selection window will be displayed:

```

D Data Field
B Block
C Constant
H Handle
S Structure
U Globally Unique ID
* Comment
    
```

Select "Data Field". The following window will be displayed:

```

..... Data Field .....
·Level: .
·Field Name: .
·Field Format: .
·Field Length: .
·Arraydefinition: .
·Edit Mask: .
·
·Header Definition: .
·
·Initialization: .
·Value Clause: .
·Comment: .
.....
    
```

Enter the following to define a variable, using the TAB key to position the cursor to the next field in the window:

```

..... Data Field .....
·Level:      1 .
·Field Name: #NAME-START .
·Field Format: A .
·Field Length: 20 .
·Arraydefinition: .
·Edit Mask: .
·
·Header Definition: .
·
·Initialization: .
·Value Clause: .
·Comment: .
.....
    
```

When you have entered all the above, press ENTER.

Then enter the following to define a second variable:

```

..... Data Field .....
·Level:          1
·Field Name:     #NAME-END
·Field Format:   A
·Field Length:  20
·Arraydefinition:
·Edit Mask:
·
·Header Definition:
·
·Initialization:
·Value Clause:
·Comment:
.....
    
```

When you have entered all the above, press ENTER.

Then define a third variable:

```

..... Data Field .....
·Level:          1
·Field Name:     #MARK
·Field Format:   A
·Field Length:   1
·Arraydefinition:
·Edit Mask:
·
·Header Definition:
·
·Initialization:
·Value Clause:
·Comment:
.....
    
```

Press ENTER.

Then press ESC. The local data area now looks as follows:

```

                                Press <ESC> to enter command mode
Mem:          Lib: SYSEXPB   Type: LOCAL   Bytes:   291  Line:    0 of:   3
C T   Comment
*     *** Top of Data Area ***
  1 #NAME-START                A    20
  1 #NAME-END                  A    20
  1 #MARK                       A     1
*     *** End of Data Area ***
    
```

Press ESC and enter "CHECK" in the command line to determine whether the data area contains errors.

### Step 3

Now, three variables that are defined in a DDM are to be included in the local data area. Press ESC to enter edit mode and then enter the command "V" as shown below. The following window will be displayed:

```

                                Press <ESC> to enter command mode
Mem:                               Lib: SYSEXP  Type: LOCAL   Bytes: 291  Line: 3 of: 3
C T L Name of Datafield             F Leng Index/Comment
*   *** Top of Data Area ***
  1 #NAME-START                      A 20
  1 #NAME-END                        A 20
V  1 #MARK                           A 1
*   *** End of Data Area ***

..... View Definition .....
•Name of View:                       .
•Name of DDM:                        .
•Comment:                             .
.....
    
```

Enter the following:

```

Name of View: PERSON-VIEW
Name of DDM:  EMPLOYEES
Comment:
    
```

Then press ENTER.

A window listing all field definitions of the DDM "EMPLOYEES" will be displayed:

```

..... DDM: EMPLOYEES .....
•  1 AA PERSONNEL-ID                A 8 D
•  G 1 AB FULL-NAME
•  2 AC FIRST-NAME                   A 20 N
•  2 AD MIDDLE-I                     A 1 N
•  2 AE NAME                         A 20 D
•  1 AD MIDDLE-NAME                  A 20 N
.....
    
```

You select the fields that are to be included in the local data area by marking them with an "X" in the left-hand column. Mark the following fields: PERSONNEL-ID, NAME, DEPT, and LEAVE-DUE. With the arrow keys, you can scroll through the DDM until these field names are shown on the list. After you have marked all four fields, press ENTER.

The data area now includes the fields selected from the DDM:

```

                                Press <ESC> to enter command mode
Mem:                               Lib: SYSEXP  Type: LOCAL   Bytes: 776  Line: 4 of: 8
C T L Name of View                   Name of DDM
  1 #NAME-START                      A 20
  1 #NAME-END                        A 20
  1 #MARK                             A 1
V  1 PERSON-VIEW                     EMPLOYEES
  2 PERSONNEL-ID                     A 8
  2 NAME                             A 20
  2 DEPT                             A 6
  2 LEAVE-DUE                        N 2.0
*   *** End of Data Area ***
    
```

The local data area is now complete. Press ESC and enter "CHECK" in the command line to check that it contains no errors.

## Step 4

A data area must be stored in compiled form before any program referencing the data area can be compiled and executed.

In the command line enter "SAVE LDA01" and press ENTER. Then store the LDA by entering "STOW" in the command line and pressing ENTER.

## Step 5

Now that the variables are defined in the local data area LDA01, the DEFINE DATA statement of the program PGM01 has to be changed from actually containing the definitions of variables to merely referencing the local data area in which the variables are defined.

In the command line of the data area editor, enter the following command:

### E PGM01

This invokes the program editor and reads the program PGM01 into the work area of the editor.

Type a "d" at the beginning of each line (over the line numbers) that defines a variable within the DEFINE DATA statement, as shown below:

d	01	#NAME-START	(A20)
d	01	#NAME-END	(A20)
d	01	#MARK	(A1)
d	01	EMPLOYEES-VIEW	VIEW OF EMPLOYEES
d	02	PERSONNEL-ID	(N8)
d	02	NAME	(A20)
d	02	DEPT	(A6)
d	02	LEAVE-DUE	(N2)

When you press ENTER, these lines are deleted from the program. The lines can also be deleted by using the F4 key and pressing ENTER.

Then enter the following after the word "LOCAL":

USING LDA01
-------------

The program should now look as shown below.

### Program PGM01:

```

* Example Program PGM01
* -----
DEFINE DATA
  LOCAL USING LDA01
END-DEFINE
*
REPEAT
*
  INPUT USING MAP 'MAP01'
*
  IF #NAME-START = '.'
    ESCAPE BOTTOM
  END-IF
*
  MOVE #NAME-START TO #NAME-END
*
  RD1. READ EMPLOYEES-VIEW
    BY NAME
    STARTING FROM #NAME-START
    ENDING AT #NAME-END
*
  IF LEAVE-DUE >= 30
    PERFORM MARK-SPECIAL-EMPLOYEES
  ELSE
    RESET #MARK
  END-IF
*
  DISPLAY NAME 3X DEPT 3X LEAVE-DUE 3X '>=30' #MARK
*
END-READ
*
IF *COUNTER (RD1.) = 0
  REINPUT 'PLEASE TRY ANOTHER NAME'
END-IF
*
END-REPEAT
*
DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
  MOVE '*' TO #MARK
END-SUBROUTINE
*
END

```

## Step 6

You can enter comments into a program to document the changes you made to the program. Comments help anyone editing or maintaining a source program, and are ignored in processing.

You mark a line as a comment line by entering either an asterisk and a blank (\*) or two asterisks (\*\*) at the beginning of the line. In the rest of the line you can then enter any comment. If you wish to append a comment to a line containing a statement, you enter the character string blank-slash-asterisk (/); anything to the right of this character string will then be considered a comment and ignored at execution.

Redisplay the beginning of PGM01 by pressing PAGEUP.

Insert a new empty line by entering an "i" in the first line of the program.

In this empty line, add some comment to indicate that this program has been updated, as shown in the example below.

**Example:**

```
* Example Program PGM01
* Program now uses a local data area
* -----
DEFINE DATA
  LOCAL USING LDA01 /* this comment is for demonstration purposes only
END-DEFINE
...
```

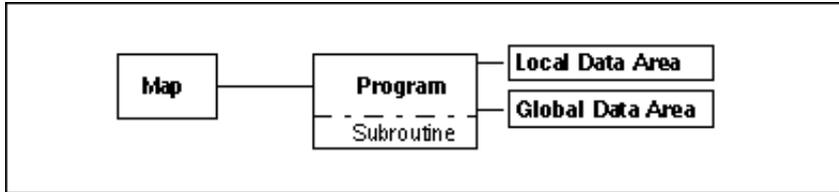
The subsequent sessions in this section show further examples of comments, which you can add to keep track of the changes you have made.

**Step 7**

CHECK the program and correct any errors. RUN the program to confirm that the results are the same as when the DEFINE DATA statement did not reference a local data area. STOW the program so that it will be available for Session 3.

End of Session 2.

## Session 3 - Creating a Global Data Area



In Natural, data can be defined in a single location outside any particular program or routine. Data defined in such a *global data area* can then be shared by multiple programs/routines.

In this session, you will do the following:

- create a global data area;
- modify the local data area created in Session 2;
- modify the program to reference not only the local data area, but also the new global data area.

### Step 1

In the command line of the program editor, enter the following command:

**E LDA01**

The data area editor will be invoked and the local data area LDA01 read into the work area of the editor.

### Step 2

To create a new data area, save the contents of the work area under a different name: In the command line of the data area editor, enter the following command:

**SAVE GDA01**

Then enter the command:

**READ GDA01**

to read the newly created data area into the work area of the editor. As it is identical to the original one, only the name at the top of the data area editor will change from "LDA01" to "GDA01".

Then enter the command:

**SET TYPE GLOBAL**

As a result, you will now be editing the data area in the editor as a *global data area*.

### Step 3

Press ESC to switch to edit mode and use the line command "d" to delete the first two lines (#NAME-START and #NAME-END). The global data area now looks as follows:

```

                                Press <ESC> to enter command mode
Mem: GDA01      Lib: SYSEXPB      Type: GLOBAL      Bytes: 993 Line: 0 of: 8
C T  Comment
*   *** Top of Data Area ***
  1 #MARK                A      1
V 1 EMPLOYEE-VIEW                EMPLOYEES
  2 PERSONNEL-ID           A      8
  2 NAME                   A     20
  2 DEPT                   A      6
  2 LEAVE-DUE              N     2.0
*   *** End of Data Area ***
    
```

Press ESC to invoke command mode.

Then enter the command STOW in the command line to store the global data area in source and object form.

### Step 4

Now some variables must be removed from the local data area, because they are now defined in the new global data area.

In the command line, enter the command "READ LDA01" to read the local data area LDA01 in the work area of the editor.

Use the "d" line command to delete from LDA01 all variables which are now also defined in GDA01; that is, everything except the two variables #NAME-START and #NAME-END. The modified local data area now looks as follows:

```

                                Press <ESC> to enter command mode
Mem: LDA01      Lib: SYSEXPB      Type: LOCAL      Bytes: 993 Line: 0 of: 8
C T  Comment
*   *** Top of Data Area ***
  1 #NAME-START           A     20
  1 #NAME-END             A     20
    
```

Press ESC and then enter the command STOW.

The modified local data area is now ready to be referenced in the program.

### Step 5

As the data to be used by the program PGM01 are now located in two data areas, the DEFINE DATA statement in the program has to be modified so that it references both the global data area GDA01 as well as the local data area LDA01.

In the command line of the data area editor, enter the command "EDIT PGM01" to invoke the program editor and read the program into the work area of the editor.

After the line containing "DEFINE DATA", insert an empty line by entering "i". In the empty line, type in:

```
GLOBAL USING GDA01
```

Then press ENTER twice.

## Step 6

Now you will change the output produced by the program: you will add a WRITE TITLE statement, and you will modify the DISPLAY statement.

The WRITE TITLE statement used in this program produces a title on multiple lines in the resulting report. The slash "/" notation causes a line advance. As nothing else is specified, the title lines will be displayed centered and not underlined.

Insert a WRITE TITLE statement above the DISPLAY statement, and change the DISPLAY statement, as shown below.

Use the "i" command to create the empty lines you need for these modifications.

Also, add some comments at the top of the program to indicate the changes you have made.

The revised program - particularly the DEFINE DATA, WRITE TITLE and DISPLAY statements blocks - should now look as shown below.

### Program PGM01:

```

* Example Program PGM01
* Program now uses a local data area and a global data area.
* WRITE TITLE has been added, and DISPLAY statement has been changed.
* -----
DEFINE DATA
  GLOBAL USING GDA01
  LOCAL USING LDA01
END-DEFINE
*
REPEAT
*
  INPUT USING MAP 'MAP01'
*
  IF #NAME-START = '.'
    ESCAPE BOTTOM
  END-IF
*
  MOVE #NAME-START TO #NAME-END
*
  RD1. READ EMPLOYEES-VIEW
        BY NAME
        STARTING FROM #NAME-START
        THRU #NAME-END
*
  IF LEAVE-DUE >= 30
    PERFORM MARK-SPECIAL-EMPLOYEES
  ELSE
    RESET #MARK
  END-IF
*
  WRITE TITLE
    / '*** PERSONS WITH 30 OR MORE DAYS LEAVE DUE ***'
    / '*** ARE MARKED WITH AN ASTERISK ***'//
*
  DISPLAY 23X '//N A M E' NAME
          3X '//DEPT' DEPT
          3X '//LV/DUE' LEAVE-DUE
          3X '//*' #MARK
*
  END-READ
*
  IF *COUNTER (RD1.) = 0
    REINPUT 'PLEASE TRY ANOTHER NAME'
  END-IF
*
  END-REPEAT
*
  DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
    MOVE '*' TO #MARK
  END-SUBROUTINE
END

```

## Step 7

Once you have made all changes, CHECK the program and correct any errors if necessary.

Then RUN the program; on the input screen, enter the name "SMITH".

Note the differences in the report output, which now looks as follows:

```

*** PERSONS WITH 30 OR MORE DAYS LEAVE DUE ***
*** ARE MARKED WITH AN ASTERISK ***

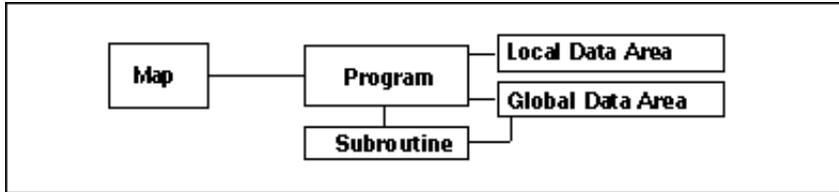
```

N A M E	DEPT	LV DUE	*
-----	-----	---	-
SMITH	SYSA05	30	*
SMITH	MGMT10	13	
SMITH	SALE40	31	*
SMITH	MGMT10	14	
SMITH	MGMT10	18	
SMITH	SALE20	36	*
SMITH	MGMT30	22	
SMITH	TECH10	38	*
SMITH	TECH10	10	
SMITH	SALE20	14	
SMITH	TECH05	20	
SMITH	MGMT10	29	
SMITH	TECH10	33	*

When the execution of the program has finished without any errors, STOW the program for the next session.

End of Session 3.

## Session 4 - Creating an External Subroutine



In Natural, a *subroutine* can be defined either within a program, or as an external subroutine outside the program.

Until now, the subroutine "MARK-SPECIAL-EMPLOYEES" has been defined within the program using a DEFINE SUBROUTINE statement. In this session, the subroutine will be defined as a separate object external to the program.

Because both internal and external subroutines are invoked with a PERFORM statement, only minimal changes to the program are required.

### Step 1

Make a copy of the program PGM01 by saving it under a different name: enter the command "SAVE SUBR01" in the command line of the program editor.

Then enter the command "READ SUBR01" to read the new copy into the work area of the editor.

Enter the command "SET TYPE S" to change the object type from *program* to *subroutine*.

Delete all lines of the subroutine except the comment lines, the DEFINE DATA and DEFINE SUBROUTINE blocks, and the END statement, so that the subroutine looks as follows:

#### Subroutine SUBR01:

```

* Example Subroutine: SUBR01
* *****
DEFINE DATA
  GLOBAL USING GDA01
  LOCAL USING LDA01
END-DEFINE
*
DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
  MOVE '*' TO #MARK
END-SUBROUTINE
END
  
```

CHECK your changes and correct any errors. Then STOW the subroutine.

### Step 2

Now that the subroutine is located in SUBR01, the internal subroutine must be removed from program PGM01.

Enter the command "EDIT PGM01".

Delete the following lines, containing the internal subroutine definition, from the program:

```

DEFINE SUBROUTINE MARK-SPECIAL-EMPLOYEES
  MOVE '*' TO #MARK
END-SUBROUTINE
  
```

The program now looks as shown below.

### Program PGM01:

```

* Example Program PGM01
* Program now uses a local data area and a global data area.
* WRITE TITLE has been added, and DISPLAY statement has been changed.
* The subroutine is now external in SUBR01.
* -----
DEFINE DATA
  GLOBAL USING GDA01
  LOCAL USING LDA01
END-DEFINE
*
REPEAT
*
  INPUT USING MAP 'MAP01'
*
  IF #NAME-START = '.'
    ESCAPE BOTTOM
  END-IF
*
  MOVE #NAME-START TO #NAME-END
*
  RD1. READ EMPLOYEES-VIEW
        BY NAME
        STARTING FROM #NAME-START
        THRU #NAME-END
*
  IF LEAVE-DUE >= 30
    PERFORM MARK-SPECIAL-EMPLOYEES
  ELSE
    RESET #MARK
  END-IF
*
  WRITE TITLE
  / '*** PERSONS WITH 30 OR MORE DAYS LEAVE DUE ***'
  / '*** ARE MARKED WITH AN ASTERISK ***'//
*
  DISPLAY 23X '//N A M E' NAME
          3X '//DEPT' DEPT
          3X '//LV/DUE' LEAVE-DUE
          3X '//*' #MARK
*
  END-READ
*
  IF *COUNTER (RD1.) = 0
    REINPUT 'PLEASE TRY ANOTHER NAME'
  END-IF
*
END-REPEAT
*
END

```

### Step 3

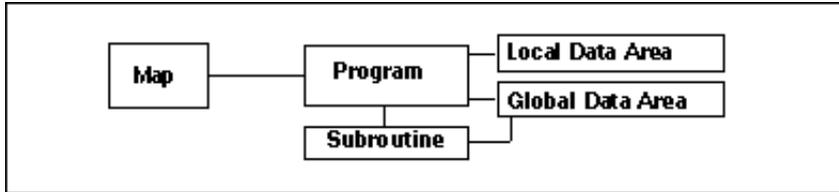
CHECK the program and correct any errors. Then RUN the program to make sure that the results are the same with an external subroutine as previously with an internal subroutine.

STOW the program for the next session.

Then leave the editor and redisplay the Natural Main Menu.

End of Session 4.

## Session 5 - Editing a Map



In this session you will edit the screen (map) prompting for an employee name. In short, you will do the following:

- add a field to enter an ending name for the range of employees;
- modify the layout of the map;
- attach a processing rule to one of the fields in the map;
- assign a help routine to the map.

### Step 1

On the Natural Main Menu, select "Direct" and enter the following command in the Direct Command window:

**EDIT MAP01**

This will invoke the Natural map editor for you to modify the map "MAP01".

On the Map Editor Menu, select "Create".

From the Create selection window, select "Text Constant".

Move the cursor to the line below the one containing "PLEASE ENTER STARTING NAME", and enter the following text:

```
PLEASE ENTER ENDING NAME :
```

Select an attribute for the text constant, and then press ESC to redisplay the Map Editor Menu.

### Step 2

Select "Create" and then "User Defined".

Position the cursor leaving one blank after the end of the text constant you just entered and press ENTER.

From the selection window of field formats, select the format "Alphanumeric".

Enter a field length of "20" in the Extended Field Editing window.

Then specify a name for the field: Type over the "#1" and enter "#NAME-END" as the name.

Use the PF2 key to select the item "AD=" from the Extended Field Editing window. The Attribute Definition window will be displayed:

```

..Attribute Definition...
·Representation      ·
·Alignment           ·
·I/O Characteristics ·
·Mandatory Characters ·
·Length Characteristics ·
·Upper/Lower Case    ·
·Filler Character     ·
.....

```

Select "I/O Characteristics". The following window will be displayed:

```

A  Input, non-protected
M  Output, modifiable
O  Output, protected

```

Select "M - Output, modifiable". Then press ENTER.

Then select "Filler Character" from the Attribute Definition window.

Enter an underscore "\_" as filler character and press ESC. Press ENTER to finish the attribute definition.

The map now looks as follows:

```

PERSONNEL INFORMATION
DD/DD/DD
TT:TT:TT
PLEASE ENTER STARTING NAME: XXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER ENDING NAME: XXXXXXXXXXXXXXXXXXXXXXXX

```

### Step 3

The next step is to change the positions of fields on the map.

Select the text constant "PERSONNEL INFORMATION" with the cursor, and press ESC.

From the Map Editor Menu, select the function "Drag".

With the arrow keys, move the text horizontally to the center of the screen.

Press ENTER to fix the text in its new position.

Then select the field \*DATX (DD/DD/DD) move it up one line.

Then move the field \*TIMX (TT:TT:TT) up one line.

The map should now look as follows:

```

DD/DD/DD                PERSONNEL INFORMATION
TT:TT:TT

PLEASE ENTER STARTING NAME: XXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER ENDING NAME: XXXXXXXXXXXXXXXXXXXXXXXX

```

Press ESC to redisplay the Map Editor Menu.

### Step 4

The next step is to create a processing rule and assign it to a map field.

**Note:**

If you use the OpenVMS TPU editor or the UNIX vi editor as your program editor, you cannot edit processing rules.

From the Map Editor, select the field #NAME-START (that is, the field after the text "PLEASE ENTER STARTING NAME"), enter modify mode, and display the Extended Field Editing window.

With the TAB key, move to "Rule Editing" and enter a "Y" (key-sensitive). The following screen will be displayed:

```

..Current Field: #NAME-START.....
.
.           R U L E   E D I T I N G (Esc = Quit)           .
·Rules                                           Fields ·
.....
    
```

Two options are available: Rules and Fields.

If Fields were selected, the fields in this map would be listed.

Select "Rules". A selection window will be displayed:

```

..Current Field: #NAME-START.....
.
.           R U L E   E D I T I N G (Esc = Quit)           .
·Rules                                           Fields ·
.....
·<CREATE>·
.....
    
```

If there were processing rules already assigned to the field, their ranks would be listed in this selection window. As no rules have yet been created for the field, the selection window only contains the item <CREATE>, which allows you to create a new processing rule.

### Step 5

Select <CREATE>. The rule editor will be invoked.

You enter the source code for a processing rule is the same way as you enter source code for a program.

At the beginning of the "top of data" line, enter the command "I" to insert an empty line.

Then enter the following processing rule:

```

IF & = ' ' REINPUT 'Please type in a name'
                MARK *&
END-IF
    
```

**Note:**

The ampersand (&) in the processing rule will be dynamically substituted by the name of the field to which the processing rule is attached.

### Step 6

Up to 100 processing rules (Rank 0 - 99) can be attached to a field. At map execution, the processing rules are executed in ascending order by rank and then by screen position of the field.

In the command line, enter the command "P=1" to assign Rank 1 to the processing rule; at the same time, the rule is saved. The rank will be displayed in the Rules selection window:

```

..Current Field: #NAME-START.....
.
.           R U L E   E D I T I N G (Esc = Quit)           .
. Rules                                           Fields .
.....
.<CREATE>.
.      1.
.....
    
```

Keep pressing ESC until the Map Editor Menu is displayed again.

### Step 7

On the Map Editor Menu, select "Ops. Map", and from the resulting selection window, select the function "Test Map".

The map (including its processing rules) will be tested, and displayed on the screen in the same way as if it were invoked from the program "PGM01".

When you press ENTER without entering anything on the screen, the processing rule will be executed:

```

Please type in a name
2001-10-25           PERSONNEL INFORMATION
13:40:34

PLEASE ENTER STARTING NAME: _____
PLEASE ENTER ENDING NAME:  _____
    
```

**Note:**

The text "Please type in a name" will be output in the message line. This line may be at the top or bottom of your screen, depending on your NATPARM setting.

In the field after "PLEASE ENTER STARTING NAME", type in a name and press ENTER.

Keep pressing ENTER until testing is finished and the map editing screen is displayed again.

Then press ENTER to redisplay the Map Editor Menu.

### Step 8

Select "Create", and from the Create selection window, select "Text Constant".

Below the line containing "PLEASE ENTER ENDING NAME", enter the following text:

```

To stop, type in:
    
```

You are then prompted to select a display attribute for the field. Select "Default".

Then redisplay the Map Editor Menu.

### Step 9

Select "Create", and then "Text Constant".

Position the cursor one blank past the text you just entered. Enter a period "." as text.

You are prompted to select a display attribute for the text. The period is to be displayed intensified when the map is executed. Therefore select the attribute "Intensified".

Then redisplay the Map Editor Menu.

### Step 10

Repeat the above two steps to create in the next line of the map the following text constant:

```
For help, type in:
```

and after that an intensified question mark "?".

The map now looks as follows:

```
DD/DD/DD                PERSONNEL INFORMATION
TT:TT:TT

PLEASE ENTER STARTING NAME XXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER ENDING NAME XXXXXXXXXXXXXXXXXXXXX
To stop, type in: .
For help, type in: ?
```

### Step 11

The next step is to change the order of entire lines of a map.

With the cursor, select the line that contains "To stop ...", and redisplay the Map Editor Menu.

Select the Lines function. The following selection window is displayed:

```
Insert After
Erase Line
Copy After
Duplicate Line
Move After
Split Line
Join Line
```

Select "Move After".

Then with the cursor you select the line after which the line is to be moved. In this case, place the cursor in the line that contains "For help ...", and press ENTER.

The map now looks as follows:

```

DD/DD/DD                PERSONNEL INFORMATION
TT:TT:TT

PLEASE ENTER STARTING NAME XXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER ENDING NAME XXXXXXXXXXXXXXXXXXXXXXXX
For help, type in: ?
To stop, type in: .
    
```

### Step 12

The next step is to insert an empty line in the map.

With the cursor, select the line that contains "PLEASE ENTER ENDING NAME", and redisplay the Map Editor Menu.

Select the Lines function, and from the selection window, select "Insert After".

An empty line will be inserted after the selected line:

```

DD/DD/DD                PERSONNEL INFORMATION
TT:TT:TT

PLEASE ENTER STARTING NAME XXXXXXXXXXXXXXXXXXXXXXXX
PLEASE ENTER ENDING NAME XXXXXXXXXXXXXXXXXXXXXXXX

For help, type in: ?
To stop, type in: .
    
```

### Step 13

The next step is to assign a helproutine to the field #NAME-START.

With the cursor, select the field #NAME-START (the field after "PLEASE ENTER STARTING NAME"), and press ENTER to redisplay the Map Editor Menu.

Select "Modify". The Extended Field Editing window for the selected field will be displayed.

With the TAB key, select "HE=". A window will be displayed for you to enter the name of a helproutine. Enter the following name (in apostrophes!):

```
'HELP01'
```

"HELP01" - which is yet to be created - is now assigned as helproutine to the field.

### Step 14

Keep pressing ENTER until the Map Editor Menu reappears.

Select "Ops. Map", and from the selection window, select the function "Stow Map". The Stow window will be displayed:

```

...Stow Map As: ....
.Name: MAP01      .
.Libr: SYSEXP    .
.....
    
```

As the map name "MAP01" is already entered, just press ENTER to stow the map.

Upon completion of the Stow function, the map is displayed in the work area of the editor.

Redisplay the Map Editor Menu.

### Step 15

Now the helproutine "HELP01" has to be created.

Select "Quit" to redisplay the Natural Main Menu.

Select "Direct", and in the Direct Command window, enter the following command to create a new map:

#### **EDIT MAP or in short form: E M**

The Map Editor Menu will be displayed. Select "Create", and then "Text Constant".

Enter the first line of the following text:

Type in the name of an employee in the first field and press ENTER. You will then receive a list of all employees of that name. If you enter names in both fields, you will get a list of all employees in alphabetical order from the first specified name to the second specified name.

Then repeat the creation of text constants, one for each line of text, until all of the above text has been entered.

### Step 16

Then redisplay to the Map Editor Menu.

Select "Ops. Map", and then "Prof. Map". The Map Settings window will be displayed:

```
.Map Settings.....
.Format                Context
-----
.Page Size .....: 23
.Line Size .....: 79      WRITE Statement ...: N
.Layout .....:
. dynamic .....: N      Help Routine .....:
.Zero Print ....: N      Help Parameter ....:
.Upper Case ....: Y      as field default .: N
.Documentation Skip ...: N      Help Text .....: N
.Decimal Char ..: .      Position Line ....: 0      Column : 0
.Standard Keys .: N      AutoRuleRank .....: 1
.Right Justify .: Y
.Print Mode ....:      Filler Characters
.-----
.Control Var ...:      Optional, Partial .: _
.                      Required, Partial .: _
.Field Sensitive: N      Optional, Complete .: _
.                      Required, Complete .: _
.....
```

The page size is set to 23, the line size to 79.

Change the page size to 15 and the line size to 25 by typing over the existing values.

Press ENTER, and you are prompted:

```
Warning Please confirm lines columns may be lost (Y/N).
```

Enter "Y" (key-sensitive), and the editor reappears. Press ENTER for the map menu and select the function "Stow Map".

In the Stow window, enter the name "HELP01" to store the new map.

Upon completion of this Stow function, the map is displayed in the work area of the editor.

Redisplay the Map Editor Menu.

Then select "Quit" to redisplay the Main Menu.

### Step 17

Now the map MAP01 has to be tested to see if the attached help map HELP01 is executed correctly.

Select "Direct" and enter the following command in the Direct Command window:

#### E MAP01

The Map Editor Menu for MAP01 will be displayed.

Select "Ops. Map", and then "Test Map".

When you enter a question mark (?) in the field after "PLEASE ENTER STARTING NAME", the help routine or help map assigned to that field should execute and the help text entered in Step 15 displayed.

Press ENTER twice, and the processing rule attached to the first field will be executed, prompting you to "Please type in a name".

Enter a name to end testing. The map editing screen will be displayed.

Redisplay the Map Editor Menu, and from there with "Quit", redisplay the Main Menu.

## Step 18

Now the program PGM01 has to be adjusted to the modified map.

Select "Direct" and enter the command "E PGM01" in the Direct Command window to read PGM01 into the work area of the program editor.

Until now, the program included the instruction:

```
MOVE #NAME-START TO #NAME-END
```

Thus the start value for the list displayed by the program was also the end value, which meant that in the example used the list contained only employees whose names were "SMITH". (Otherwise, all employees from "SMITH" to the end of the alphabet would have been included in the report.) Now the map allows you to specify both a start value *and* an end value for the list to be output. However, an IF statement has to be added to the program to handle a situation in which no end value is specified.

Change the program so that the MOVE statement is contained within an IF statement block as follows:

```
IF #NAME-END = ' '  
  MOVE #NAME-START TO #NAME-END  
END-IF
```

The program should now look as shown below.

### Program PGM01:

```

* Example Program PGM01
* Program now uses a local data area and a global data area.
* WRITE TITLE has been added, and DISPLAY statement has been changed.
* The subroutine is now external in SUBR01.
* -----
DEFINE DATA
  GLOBAL USING GDA01
  LOCAL USING LDA01
END-DEFINE
*
REPEAT
*
  INPUT USING MAP 'MAP01'
*
  IF #NAME-START = '.'
    ESCAPE BOTTOM
  END-IF
*
  IF #NAME-END = ' '
    MOVE #NAME-START TO #NAME-END
  END-IF
*
  RD1. READ EMPLOYEES-VIEW BY NAME
        STARTING FROM #NAME-START
        ENDING AT #NAME-END
*
    IF LEAVE-DUE >= 30
      PERFORM MARK-SPECIAL-EMPLOYEES
    ELSE
      RESET #MARK
    END-IF
*
    WRITE TITLE
    / '*** PERSONS WITH 30 OR MORE DAYS LEAVE DUE ***'
    / '*** ARE MARKED WITH AN ASTERISK ***'//
*
    DISPLAY 23X '//NAME' NAME
            3X '//DEPT' DEPT
            3X '//LV/DUE' LEAVE-DUE
            3X '//*' #MARK
*
  END-READ
*
  IF *COUNTER (RD1.) = 0
    REINPUT 'PLEASE TRY ANOTHER NAME'
  END-IF
*
END-REPEAT
*
END

```

## Step 19

CHECK the program and correct any errors that may be indicated.

Then RUN the program. On the input screen, enter the names "JONES" and "JUNG" as start value and end value respectively.

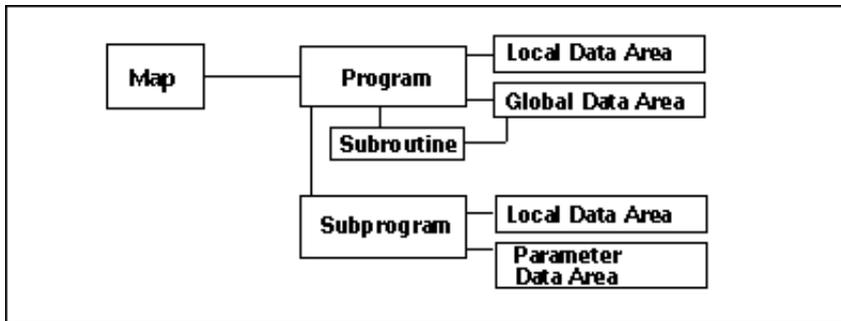
Then type "." in the first field and press ENTER to stop the execution of the program.

After the program has been executed, STOW it for the next session.

Then redisplay the Natural Main Menu.

End of Session 5.

## Session 6 - Invoking a Subprogram



In Natural, both *subprograms* and *subroutines* can be invoked from a program. They differ from one another in the way data from the invoking program can be accessed.

As shown in Session 4, a *subroutine* can access the same global data area as the invoking program. However, a *subprogram* is invoked with a CALLNAT statement, and with this statement, parameters can be passed from the invoking program to the subprogram. These parameters are the only data available to the subprogram from the invoking program.

The passed parameters must be defined either within the DEFINE DATA PARAMETER statement of the subprogram, or in a parameter data area used by the subprogram.

In addition, a subprogram can have its own local data area, in which the fields to be used within the subprogram are defined.

In this session, you will create a subprogram with a parameter data area and a local data area. Moreover, a CALLNAT statement to invoke the subprogram will be added to the main program; also the main program's local data area has to be modified. In the subprogram, the employees selected by the main program will be the basis to select the corresponding vehicle information from the VEHICLES file. As a result, the report will contain vehicle information as well as employee information.

### Step 1

First, the main program's local data area has to be modified.

On the Main Menu, select "Direct" and enter the command "E LDA01" in the Direct Command window.

The data area editor will be invoked and the local data area LDA01 will be read into the work area:

```

                                Press <ESC> to enter command mode
Mem: LDA01      Lib: SYSEXPB   Type: LOCAL      Bytes:  639  Line:   0 of:   5
C T  Comment
*   *** Top of Data Area ***
  1 #NAME-START                A   20
  1 #NAME-END                  A   20

*   *** End of Data Area ***
    
```

Move the cursor down to highlight "#NAME-END" and type "i" to insert a line. Add the variables #PERS-ID, #MAKE and #MODEL to the local data area, as shown below. They will be used in the CALLNAT statement to be added to the program.

The local data area now looks as follows:

```

                                Press <ESC> to enter command mode
Mem: LDA01      Lib: SYSEXPB   Type: LOCAL      Bytes: 639 Line: 0 of: 5
C T  Comment
*   *** Top of Data Area ***
  1 #NAME-START                A   20
  1 #NAME-END                  A   20
  1 #PERS-ID                    A    8
  1 #MAKE                       A   20
  1 #MODEL                      A   20
*   *** End of Data Area ***
    
```

CHECK the local data area and correct any errors. Then STOW it.

### Step 2

A parameter data area is needed for the subprogram. With minor modifications it is possible to use the local data area LDA01 to create this parameter data area.

Make a copy of LDA01 by saving it under a different name: in the command line of the editor, enter the command "SAVE PDA02".

Then enter the command "READ PDA02" to read the new copy into the editor work area.

Then enter the command "SET TYPE PARAMETER" to change the data area type from "LOCAL" to "PARAMETER".

With the line command "d", delete the first two lines. The parameter data area now looks as follows:

```

                                Press <ESC> to enter command mode
Mem: PDA02      Lib: SYSEXPB   Type: PARAMETER Bytes: 445 Line: 1 of: 3
C T L Name of Datafield          F Leng Index/Comment          M
*   *** Top of Data Area ***
  1 #PERS-ID                    A    8
  1 #MAKE                       A   20
  1 #MODEL                      A   20
*   *** End of Data Area ***
    
```

CHECK the parameter data area and correct any errors. Then STOW it.

### Step 3

Now, a local data area for the subprogram has to be created.

In the command line of the editor, enter the command "CLEAR" to clear the work area.

Then enter the command "SET TYPE LOCAL" to change the data area type.

Switch to edit mode by pressing ESC and then enter a "V" in the first column of the screen.

The View Definition screen will be displayed. Enter the following:

```

..... View Definition .....
·Name of View: CAR-VIEW      ·
·Name of DDM:  VEHICLES   ·
·Comment:      ·
.....
    
```

Press ENTER. A window listing all field definitions of the DDM "VEHICLES" will be displayed.

Select the fields that are to be included in the data area by marking them with an "X" in the left-hand column. Mark the following fields: PERSONNEL-ID, MAKE and MODEL After you have marked all three fields, press ENTER.

The local data area now looks as follows:

```

                                Press <ESC> to enter command mode
Mem:                               Lib: SYSEXP  Type: LOCAL      Bytes: 632  Line: 0 of: 5
C T  Comment
*   *** Top of Data Area ***
V 1  CAR-VIEW                               VEHICLES
   2  PERSONNEL-ID                          A    8
   2  MAKE                                   A   20
   2  MODEL                                  A   20
*   *** End of Data Area ***
    
```

CHECK the local data area and correct any errors.

Then stow it by entering the command "STOW LDA02".

Redisplay the Natural Main Menu.

### Step 4

The next step is to create the subprogram itself.

Invoke the program editor with the EDIT command, write the subprogram shown below, and STOW it under the name "SPGM02".

#### Subprogram SPGM02:

```

* Example Subprogram 'SPGM02'
* *****
DEFINE DATA
  PARAMETER USING PDA02
  LOCAL      USING LDA02
END-DEFINE
*
FD1. FIND (1) CAR-VIEW WITH PERSONNEL-ID = #PERS-ID
      MOVE MAKE (FD1.)   TO #MAKE
      MOVE MODEL (FD1.) TO #MODEL
      ESCAPE BOTTOM
END-FIND
END
    
```

This subprogram receives the personnel number passed by the main program and then uses this number as the basis of a search of the VEHICLES file.

### Step 5

Now, the main program PGM01 has to be modified to invoke the subprogram.

Read the program into the editor, and insert the following statements immediately before the WRITE TITLE statement:

```
RESET #MAKE #MODEL  
CALLNAT 'SPGM02' PERSONNEL-ID #MAKE #MODEL
```

Some of the parameters passed in the CALLNAT statement are defined in the global data area, and some in the local data area.

Please note that the variables defined in the parameter data area of the subprogram need not have the same name as the variables in the CALLNAT statement; it is only necessary that they match in sequence, format, and length.

As the program receives vehicle information from the subprogram, which is also to be output in the report, the DISPLAY statement has to be modified as shown below.

The program should then look as follows:

### **Program PGM01:**

```

* Example Program PGM01
* Program now uses a local data area and a global data area.
* WRITE TITLE has been added, and DISPLAY statement has been changed.
* The subroutine is now external in SUBR01.
* A subprogram provides vehicle information.
* -----
DEFINE DATA
  GLOBAL USING GDA01
  LOCAL USING LDA01
END-DEFINE
*
REPEAT
*
  INPUT USING MAP 'MAP01'
*
  IF #NAME-START = '.'
    ESCAPE BOTTOM
  END-IF
*
  IF #NAME-END = ' '
    MOVE #NAME-START TO #NAME-END
  END-IF
*
  RD1. READ PERSON-VIEW BY NAME
        STARTING FROM #NAME-START
        ENDING AT #NAME-END
*
    IF LEAVE-DUE >= 30
      PERFORM MARK-SPECIAL-EMPLOYEES
    ELSE
      RESET #MARK
    END-IF
*
  RESET #MAKE #MODEL
  CALLNAT 'SPGM02' PERSONNEL-ID #MAKE #MODEL
*
  WRITE TITLE
    / '*** PERSONS WITH 30 OR MORE DAYS LEAVE DUE ***'
    / '*** ARE MARKED WITH AN ASTERISK ***'//
*
  DISPLAY      '//NAME' NAME
              2X '//DEPT' DEPT
              2X '//LV/DUE' LEAVE-DUE
              ' ' #MARK
              2X '//MAKE' #MAKE
              2X '//MODEL' #MODEL
*
  END-READ
*
  IF *COUNTER (RD1.) = 0
    REINPUT 'PLEASE TRY ANOTHER NAME'
  END-IF
END-REPEAT
END

```

## Step 6

After you have made all changes to the program, CHECK it and correct any errors; then RUN it.

Then STOW it for future reference.

End of Session 6.

# DDM Services

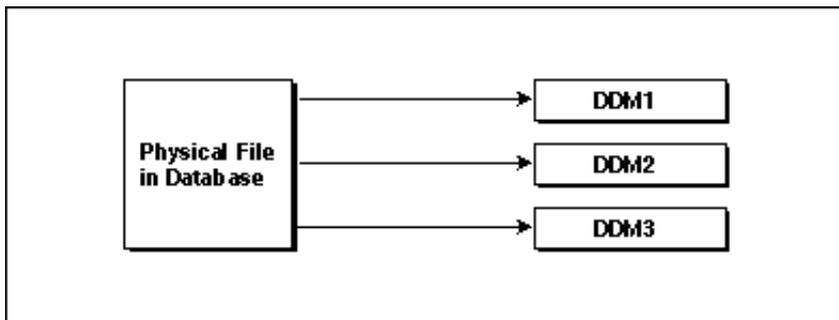
- DDMs (Data Definition Modules)
- Invoking DDM Services
- Accessing Libraries
- Creating DDMs
  - Data Conversion
- Maintaining DDMs
  - Copying DDMs
  - Deleting DDMs
  - Editing DDMs
  - Listing DDMs
  - Renaming DDMs
- Services Profile
  - Function Keys
  - Other Definitions

## DDMs (Data Definition Modules)

For Natural to be able to access a database file, a logical definition of the physical database file is required. Such a logical file definition is called a *DDM* (data definition module).

The DDM contains information about the individual fields of the file - information which is relevant for the use of these fields in a Natural program. A DDM constitutes a logical view of a physical database file.

For each physical file of a database, one or more DDMs can be defined.



The Natural Programming Guide describes the use of DDMs in more detail.

A DDM must be created and cataloged in order for the corresponding file to be available for a Natural program. The DDM Services function is used to create and maintain DDMs.

## Invoking DDM Services

To invoke DDM Services, select "Services" on the Natural Main Menu or enter SYSDDM in the direct command window.

A selection window is then displayed, from which you select "DDM Services".

The DDM Services menu is then displayed:

```
2001-10-25          DDM Services          Library : SYSTEM
14:26:25           V 5.1.1 Software AG 2001 DBID   :
User : SAG                FNR      :
.....
·  Library      DDM Maintenance      Services Profile      Quit      ·
.....
```

Selecting "Quit" terminates the DDM Services and returns you to the Natural Main Menu.

The remainder of this section is broken up into the following sections:

- Accessing Libraries
- Creating DDMs
- Maintaining DDMs
- Services Profile.

## Accessing Libraries

Just like Natural objects, DDMs are contained in libraries. To create a new DDM or maintain an existing one, you must first log on to the library where the new DDM is to be created or where a DDM already exists.

To do so, you select "Library" on the DDM Services menu. A list of all available libraries is then displayed:

```

2001-10-25                DDM Services                Library : SYSTEM
14:29:06                  V 5.1.1 Software AG 2001    DBID   :
User : SAG                 FNR                   :
.....
.  Library                DDM Maintenance          Services Profile          Quit      .
.....
.....
.  <LOGON>                .
.  DEMO                   .
.  DEMO-1                 .
.  ORD-EXAM               .
.  PRE                    .
.  SYSEXP                 .
.  SYSEXRM                .
.  SYSTEM                 .
.....

Log on to DDM library
    
```

There are three ways to proceed:

- select the desired library from the list (cursor selection);
- enter a library's first letter to display only libraries beginning with this letter and then select the desired library from the list;
- select <LOGON> and then enter the name of the desired library.

# Creating DDMs

There are two ways to create a new DDM:

- copy an existing DDM to another name;
- add a new DDM.

This section describes how to add a new DDM. For instructions on how to copy an existing DDM, see Copying DDMs.

To create a new DDM, first ensure that you are logged on to the correct library (Accessing Libraries). Select "DDM Maintenance" on the DDM Services menu. A list is displayed containing all available DDMs in the selected library:

```

2001-10-25          DDM Services          Library : SYSTEM
14:35:57           V 5.1.1 Software AG 2001 DBID   :
User : SAG          FNR                   :
.....
·  Libraries        DDM Maintenance        Services Profile    Quit
.....
                .....
                · <CREATE>                ·
                · ACTIO                    ·
                · AEH-BEDIENSTETER        ·
                · AEH-HDAT                 ·
                · EMPLOYEES                ·
                · FUNC                     ·
                · MAP                      ·
                · OBJ                      ·
                · PERSONNEL                ·
                · VEHICLES                 ·
                .....

Select DDM
    
```

Select <CREATE> from the top of the list. The following window is displayed:

```

2001-10-25          DDM Services          Library : SYSTEM
14:39:57           V 5.1.1 Software AG 2001 DBID   :
User: SAG          FNR                   :
.....
·  Library          DDM Maintenance        Services Profile    Quit   ·
.....
                .....
                · <CREATE>                ·
                · EMPLOYEES                ·
                ..... Select Database .....
                · DBID      1              ·
                .....
                Func
                Map
    
```

In this window, you enter the database ID (0 - 65535, except 255) of the file/table for which a DDM is to be created. If "0" is specified, the database ID specified in the UDB parameter of the Natural parameter module is used.

If the specified DBID identifies an Adabas database, the following window is displayed:

```

2001-10-25          DDM Services          Library: SYSTEM
14:47:12           V 5.1.1 Software AG 2001 DBID  :
User: SAG          FNR  :
.....
.  Library          DDM Maintenance        Services Profile      Quit  .
.....
.  <CREATE>        .
.  EMPLOYEES      .
.....
..... Create Adabas DDM .....
.  FNR            1                          .
.  DDM Name      .
.....
Map
    
```

In this window, you enter the file number (1 - 5000) of the file for which a DDM is to be created and the name to be assigned to the DDM.

If a database file exists, it will be read into the DDM editing area. You can then edit the DDM (as described under Editing DDMs).

**Note:**

If the corresponding database is not active, an appropriate error message is issued, but you can still read the DDM into the editing area by pressing ENTER.

If the specified DBID identifies an SQL database, the following window is displayed:

```

2001-10-25          DDM Services          Library: SYSTEM
14:56:08           V 5.1.1 Software AG 2001 DBID  :
User: SAG          FNR  :
.....
.  Library          DDM Maintenance        Services Profile      Quit  .
.....
.  <CREATE>        .
.  EMPLOYEES      .
.  EMPLTEST      .
.....
.....Select SQL Table.....
.  Table Owner:*  .
.  Table Name  :* .
.....
    
```

In this window, you enter the owner and name of the table for which a DDM is to be created.

If such a table exists, a DDM is created from this table. If no such table exists or if you specify an asterisk (\*) as table owner and/or table name, a selection list is displayed from which you can select the desired SQL table.

```

2001-10-25          DDM Services          Library: SYSTEM
15:02:23           V 5.1.1 Software AG 2001 DBID  :
User: SAG          FNR  :
.....
·  Library          DDM Maintenance        Services Profile      Quit      ·
.....
..... Import SQL Table Contents .....
·  Table Owner          Table Name          Type      ·
·  =====
·  DEMO                 AUTOMOBILES
·  DEMO                 ENMPLOYEES
·  DEMO                 SALARY
·  SAG                  ALLDATA
·  SAG                  TEST

```

A DDM is generated from the selected table and read into the DDM editing area. The DDM name corresponds to the combination of table owner and table name, for example: SAG-TEST.

**Note:**

With the Natural program DDMGEN you can generate several DDMs in a particular library without using the editor.

**Data-Type Conversion**

When a Natural program accesses data in any database, RDBMS-specific data types are converted to Natural data formats, and vice versa. The tables in this section show how Natural data formats correspond to data types in the following RDBMSs:

- Adabas
- Adabas D
- Adabas SQL Server
- DB2
- INFORMIX
- INGRES
- ORACLE
- SYBASE and Microsoft SQL Server

**Adabas**

Adabas Data Type	Natural Format/Length
A (n) alphanumeric	A (n)
B (n) binary	B (n)
F (n) fixed but: F8	I (n) I4
G (n) float	F (n)
P (n) packed	P (2 x n - 1)
U (n) unpacked	N (n)

**Adabas D**

<b>RDBMS Data Type</b>	<b>Natural Format/Length</b>
boolean	L
char ( <i>n</i> )	<i>An</i>
date	A10
fixed ( <i>p,q</i> )	<i>Np-q,q</i>
float	F8
integer	I4
long	A, DYNAMIC
long varchar	A, DYNAMIC
smallint	I2
string	<i>An</i>
time	A8
timestamp	A26
varchar	<i>An</i>

**Adabas SQL Server**

<b>RDBMS Data Type</b>	<b>Natural Format/Length</b>
char(5)	A5
char(253)	A253
decimal(5)	N5
decimal(10.4)	N(6.4)
double precision	N(10.6)
float(1...21)	N(2.6)
float(22...53)	N(10.6)
integer	I4
numeric(5)	N5
numeric(10.4)	N(6.4)
real	F4
smallint	I2

**DB2**

<b>RDBMS Data Type</b>	<b>Natural Format/Length</b>
date	A10
decimal(5)	N5
decimal(10.4)	N(6.4)
fixed character(5)	A5
float	F $n$
graphic	2* $A_n$
longvar	A, DYNAMIC
longvarg	A, DYNAMIC
large integer	I4
scientific notation	N(10.6)
small integer	I2
special data	A253
system date and time	A10
time	A8
timestmp	A26
varchar	$A_n$
varg	2* $A_n$

## INFORMIX

<b>RDBMS Data Type</b>	<b>Natural Format/Length</b>
byte	A56
char( $n$ )	$A_n$
date	A10
datetime	A26
decimal( $p,q$ )	$N_{p-q,q}$
float	F8
integer	I4
interval	A17
money	N(14.2)
real	F4
serial	I4
smallfloat	F4
smallint	I2
text	$A_n$
varchar( $n$ )	$A_n$

**INGRES**

<b>RDBMS Data Type</b>	<b>Natural Format/Length</b>
byte varying	B <i>n</i>
c( <i>n</i> )	A <i>n</i>
char ( <i>n</i> )	A <i>n</i>
date	A10
double precision	F8
float	F8
float4	F4
integer	I4
integer1	I1
long byte	B, DYNAMIC
long varchar	A, DYNAMIC
money	N(12.2)
object_key	B16
real	F4
smallint	I2
table_key	B8
text ( <i>n</i> )	A <i>n</i>
varchar ( <i>n</i> )	A <i>n</i>

**ORACLE**

<b>RDBMS Data Type</b>	<b>Natural Format/Length</b>
char ( <i>n</i> )	<i>An</i>
date	A10
decimal ( <i>p,q</i> )	<i>Np-q,q</i>
double precision	F8
float	F4
integer	I4
long	A, DYNAMIC
long raw	B, DYNAMIC
number	<i>Nn</i>
nvarchar2	<i>An</i>
raw ( <i>n</i> )	<i>Bn</i>
real	F4
rowid	<i>An</i>
smallint	I2
varchar	<i>An</i>
varchar2 ( <i>n</i> )	<i>An</i>

**SYBASE and Microsoft SQL Server**

<b>RDBMS Data Type</b>	<b>Natural Format/Length</b>
binary ( <i>n</i> )	B <i>n</i>
bit	N1
char ( <i>n</i> )	A <i>n</i>
datetime	A26
float	F8
image	B126
int	I4
money	N(15.4)
nchar ( <i>n</i> )	A <i>n</i>
nvarchar ( <i>n</i> )	A <i>n</i>
real	F4
smalldatetime	A26
smallint	I2
smallmoney	N(6.4)
text	A <i>n</i>
timestamp	B8
tinyint	I2
varbinary ( <i>n</i> )	B <i>n</i>
varchar ( <i>n</i> )	A <i>n</i>

If the length of the database column exceeds the maximum length of the corresponding Natural format, a multiple field is generated.



## Editing DDMs

To modify a DDM, enter "E" next to the DDM to be edited. The DDM will then be read into the work area, where you can edit it by using the DDM editor.

### DDM Editor

The DDM editor is used to edit the DDM currently located in the work area.

When the specified database is active and a database file with the specified file number exists, the contents of this database file are loaded in the work area and displayed on the screen; if not, an empty screen is displayed.

```

2001-10-25                DDM Services
15:30:47                  V 5.1.1 Software AG 2001                Line: 1
DBID: 1                   FNR: 1                DDM: EMPLOYEES-FILE    DEF.SEQ.:
  C   T                   L Name                   F Length S D
                        1 PERSONNEL-ID             A      8   D
                        G   1 FULL-NAME
                        2 FIRST-NAME                A     20 N
                        2 MIDDLE-NAME               A     20 N
                        2 NAME                      A     20   D
                        1 MAR-STAT                   A      1 F
                        1 SEX                       A      1 F
                        1 BIRTH                      D      6   D
                        G   1 FULL-ADDRESS
                        M   2 ADDRESS-LINE            A     20 N
                        2 CITY                       A     20 N D
                        2 POST-CODE                   A     10 N
                        2 COUNTRY                     A      3 N
                        G   1 TELEPHONE
                        2 AREA-CODE                  A      6 N
                        2 PHONE                      A     15 N
                        1 DEPT                        A      6   D
                        1 JOB-TITLE                   A     25 N D
                        P   1 INCOME
                        F1 Help F2 Choice F3 Stow+Exit F10 Stow F11 Check
                        F12 DB-Short-Names F13 Modify Header F14 Show Ext Field F15
    
```

Press F1 to display the editor help information: Press F1 once to invoke the field help key settings; press F1 twice to invoke the editor help key settings.

### Header of the DDM Editor

The header of the DDM editor displays the following information:

Field Name	Description
<b>DBID</b>	The database where the file is stored.
<b>FNR</b>	The file number.
<b>DDM</b>	The name of the DDM.
<b>Line</b>	The current edit line number.
<b>Default Sequence</b>	Logical database short name.

## Attributes of a DDM

The DDM itself comprises field-definition attributes which can be entered or modified. They are described in turn below.

### C Attribute of a DDM

Command option:

Value	Description
<i>blank</i>	No command
<b>D</b>	Delete line
<b>I</b>	Insert line
<b>X</b>	Start mark block
<b>Y</b>	End mark block
<b>H</b>	Hide block
<b>C</b>	Copy marked block
<b>M</b>	Move marked block

### T Attribute of a DDM

Field type:

Value	Description
<i>blank</i>	Elementary field
<b>G</b>	Group header
<b>M</b>	Multiple-value field
<b>P</b>	Periodic group header
*	Comment line

### DB Attribute of a DDM

The Adabas short name, which can be used to reference the field.

This name can be edited. The Adabas short names list can be displayed and/or hidden using F12 (toggle switch).

#### Note:

If you create a new DDM field and the database short names are switched OFF, the editor generates a new "unused" short name for the field; this means that for this field there is no link between the database file and the DDM. Therefore, if you have to create a new DDM field, use the INSERT command to link the new field to the database file.

**Note for ENTIRE DB:**

The short names represent 5-digit ISNs, which can be displayed but cannot be modified.

**L Attribute of a DDM**

Level number assigned to the field. Valid level numbers are 1 - 7. Level numbers must be specified in consecutive ascending order.

**Name Attribute of a DDM**

A user-defined field name (3 - 32 characters). This is the field name used in Natural programs to reference the field.

**Note:**

In SQL DDMs, the field name can be from 1 to 32 characters.

**Format and Length Attributes of a DDM**

The format and length of the field:

Value	Description
<i>blank</i>	Group
<b>A</b>	Alphanumeric (maximum length 253)
<b>B</b>	Binary (maximum length 126)
<b>D</b>	Date
<b>F</b>	Floating point (4 or 8 bytes)
<b>I</b>	Integer (1, 2 or 4 bytes)
<b>L</b>	Logical
<b>N</b>	Unpacked (maximum length 29)
<b>P</b>	Packed (maximum length 29)
<b>T</b>	Time

The standard length of a field can be overridden in a Natural program. For numeric fields (format N), the length is specified as "*nn.m*", where "*nn*" represents the number of digits before the decimal point and "*m*" represents the number of digits after the decimal point.

**Only for SQL DDMs:** In the length input field, you can specify either the field length as a numeric value or enter the keyword "DYNAMIC" to specify that the field length is variable.

For further information, see Large and Dynamic Variables/Fields.

**S Attribute of a DDM**

Null value suppression:

Value	Description
<i>blank</i>	Normal Adabas compression
<b>F</b>	Indicates that the field is defined with the Adabas fixed storage option; that is, no suppression takes place.
<b>L</b>	Indicates that a linkage attribute is defined for this field (applies to ENTIRE DB only)
<b>N</b>	Indicates that the field is defined with the Adabas null value suppression option. This means that null values for the field are not stored in the inverted list and will not be returned when the field is used to construct a basic search criteria (WITH clause of a FIND statement), in a HISTOGRAM statement, or in a READ LOGICAL statement.
<b>M</b>	Indicates that the field is defined with the SQL null-value option "not null". The Remarks column for this field contains "NN NC" (not null, not counted). Below this field, the corresponding null-indicator field is listed.

## D Attribute of a DDM

Descriptor:

Value	Description
<i>blank</i>	No Adabas descriptor is to be used.
<b>D</b>	The field is an Adabas descriptor. Value lists are created and maintained for this field by Adabas, enabling this field to be used as a search criterion in a FIND statement, as a sort key in a FIND statement, or to control logical sequential reading in a READ statement.
<b>H</b>	The field is an Adabas hyperdescriptor. A hyperdescriptor is based on an Adabas user exit; for Natural, it provides the same functionality as a phonetic descriptor (see below).
<b>N</b>	Indicates that the field is defined as a non-descriptor. A non-descriptor is not a descriptor, but can be used as a search field for a so-called non-descriptor search.
<b>P</b>	The field is an Adabas phonetic descriptor. A phonetic descriptor allows the user to perform a phonetic search on a field (for example, a person's name). A phonetic search results in the return of all values which sound similar to the search value.
<b>S</b>	The field is an Adabas subdescriptor or superdescriptor. If you select the "S", information about the contents of the superdescriptor or subdescriptor is displayed.

## Editor Commands

Editor commands can be issued either from the menu bar by selecting "Commands" or else via F key.

Several editor commands are available via function keys. To display the function-key assignments, you press F1 twice.

The following function-key assignments apply:

Key	Function
F1	Displays field-level help information. Press F1 again to display general help information.
F2	Produces a selection list for a field. You can then select one of the values listed, or you can enter the first character of an entry.
F11	Checks the DDM for duplicate names and valid field entries.
F12	Switches the display of Adabas field short names on or off (toggle switch).
F13	Changes the DBID and file number (FNR) of the DDM (shown at the top of screen, default sequence).
F14	Invokes extended field editing. Press F15 to edit information contained in the resulting window.
F15	Edits information contained in extended field editing window.

After editing is completed, you press F3 or F10 to stow (that is, save *and* catalog) the DDM.

### Extended Field Editing

The DDM editor can also be used to enter/modify field definitions.

The extended editing mode is used to specify default options for field headers and edit masks as well as remarks to be applied when the field is used in a DISPLAY or INPUT statement.

The extended editing mode is invoked by pressing F14. When you press F15, you can edit the information contained in the resulting window; F14 has a toggle effect (ON/OFF):

```

2001-10-25                DDM Services
15:38:07                  5.1.1 Software AG 2001                Line: 1
DBID: 1  FNR: 1  DDM: EMPLOYEES-FILE                DEF.SEQ.:
  C  T      L Name                F  Length  S  D
      1 PERSONNEL-ID                A      8      D
  G      1 FULL-NAME
      2 FIRST-NAME                A     20  N
      2 MIDDLE-NAME                A     20  N
      2 NAME                A     20  D
      1 MAR-STAT                A      1  F
      1 SEX                A      1  F
      1 BIRTH                D      6      D
  G      1 FULL-ADDRESS
  M      2 ADDRESS-LINE                A     20  N
      2 CITY                A     20  N  D
      2 POST-CODE                A     10  N
      2 COUNTRY                A      3  N
  G      1 TELEPHONE
      2 AREA-CODE                A      6  N
----- Extended Field Information -----
Header   : PERSONNEL-ID
Edit Mask:
Remark   :
F1 Help   F2 Choice  F3 Exit   F4 Delete  F5 Insert  F6 Mark
F7       F8 Copy    F9 Move   F10 Stow  F11 Check  F12 Quit
    
```

The Extended Field Editing information is displayed at the bottom of the screen; it changes, based on the current edit line.

**Note for ENTIRE DB:**

In addition to Header, Edit Mask and Remark, Linkage information is displayed.

In extended editing mode, the highlighted line can be moved by using UP ARROW and DOWN ARROW keys. The TAB key can be used to move the line downwards.

The extended field editing screen is terminated when ENTER is pressed with or without any modifications having been made.

### Leaving the DDM Editor

Pressing ESC invokes the following screen:

Commands		Misc	Quit	
15:44:25		V 5.1.1 Software AG 2001	Line: 1	
DBID: 1 FNR: 1 DDM: EMPLOYEES-FILE			DEF.SEQ.:	
C	T	L Name	F	Length S D
		1 PERSONNEL-ID	A	8 D
	G	1 FULL-NAME		
		2 FIRST-NAME	A	20 N
		2 MIDDLE-NAME	A	20 N
		2 NAME	A	20 D
		1 MAR-STAT	A	1 F
		1 SEX	A	1 F
		1 BIRTH	D	6 D
	G	1 FULL-ADDRESS		
	M	2 ADDRESS-LINE	A	20 N
		2 CITY	A	20 N D
		2 POST-CODE	A	10 N
		2 COUNTRY	A	3 N
	G	1 TELEPHONE		
		2 AREA-CODE	A	6 N
		2 PHONE	A	15 N
		1 DEPT	A	6 D
		1 JOB-TITLE	A	25 N D
	P	1 INCOME		
F1 Help		F2 Choice	F3 Stow+Exit	F10 Stow
F11 Check		Select a Command		

### Commands

If you select "Commands" and press ENTER, the following options appear:

Command	Function
<b>CATALOG</b>	Stores the file description in the database in object form.
<b>CHECK</b>	Checks the syntax for correctness.
<b>SAVE</b>	Stores the file description in the database in source form.
<b>SCAN</b>	Scans for a field name within the file description.
<b>STOW</b>	Stores the file description in the database in both source and object form.

### Misc

If you select "Misc" and press ENTER, the following options appear:

Option	Function
<b>DB Short Names (ON/OFF)</b>	If you select this option and press ENTER, a toggle function is invoked, which shows the DB short names which were not displayed previously and vice versa.
<b>Modify DDM Header</b>	Enables modification of DDM editor header information.
<b>Show Extended Field</b>	Displays extended field window.
<b>Edit Extended Field</b>	Enables editing of extended field window.
<b>Show Coupled Files</b>	If you select this option, all files physically coupled to the displayed DDM are listed together with the short names of the descriptors used for coupling.

### Quit

If you select "Quit" and press ENTER, the following two options appear:

Option	Function
<b>EXIT</b>	Stows the DDM before leaving the DDM editor.
<b>QUIT</b>	Leaves the DDM editor without stowing the DDM.

### Listing DDMs

To display a DDM, enter "L" next to the DDM to be listed. The selected DDM is then displayed as shown in the example below:

```

.....List: EMPLOYEES.....
. 0001 DB: 001 FILE: 001 - EMPLOYEES                                DEFAULT SEQUENCE.
. 0002
. 0003 T L DB Name                                               F Leng  S D Remark
. 0004 - - -- -----
. 0005 1 AA PERSONNEL-ID                                         A    8    D
. 0006 HD=PERSONNEL/ID
. 0007 * CNNNNNNN
. 0008 * C=COUNTRY
. 0009 G 1 AB FULL-NAME
. 0010 2 AC FIRST-NAME                                           A   20   N
. 0011 2 AD MIDDLE-I                                             A    1   N
. 0012 2 AE NAME                                                 A   20   D
. 0013 1 AD MIDDLE-NAME                                          A   20   N
. 0014 1 AF MAR-STAT                                             A    1   F
. 0015 HD=MARITAL/STATUS
. 0016 * M=MARRIED
. 0017 * S=SINGLE
. 0018 * D=DIVORCED
.....
    
```

## Renaming DDMs

To rename a DDM, enter "N" next to the DDM the be renamed. A window is then displayed in which you enter the new name of the DDM:

```
2001-10-25          DDM Services          Library : SYSTEM
15:53:14           V 5.1.1 Software AG 2001 DBID   : 1
User : SAG                                     FNR    : 1
.....
·  Library          DDM Maintenance        Services Profile      Quit      ·
.....
          .....
          · <CREATE> ·
          · N EMPLOYEES ·
          · PERSONNEL ·
          · VEHICLES ·
          ..... Rename DDM to .....
          ·
          .....
Enter new DDM name
```

**Note:**  
XREF data are taken into account.

## Services Profile

When you select "Services Profile", the following window is displayed:

```

2001-10-25          DDM Services          Library : SYSTEM
16:03:10           V 5.1.1 Software AG 2001 DBID   :
User : SAG          FNR                   :
.....
.  Library          DDM Maintenance        Services Profile      Quit  .
.....
. Function Keys .
. Other Definitions .
.....
    
```

## Function Keys

With this function, you can assign DDM-editor functions to other function keys.

When you select "Function Keys", the following window is displayed, showing the current function-key assignments:

```

..... Profile settings .....
. Delete           F4           .
. Insert           F5           .
. Mark Block       F6           .
. Unmark Block     F7           .
. Copy             F8           .
. Move             F9           .
.....
    
```

You assign a function to another key as follows:

1. In the above window, you first select the function key to which the function is assigned.
2. Then you *press* the function key to which you wish the function to be assigned.

**Note:**

It is not possible to assign a function to a key which already has a function assigned to it.

## Other Definitions

When you select "Other Definitions", a window is displayed in which you enter "Y" or "N" to determine whether or not the 2-character database short names of the DDM fields are to be displayed in the DDM editor:

**Note:**

Short names can also be displayed from the editor using F12.

```

..... Profile settings .....
. Display Database short names N .
.....
    
```

# Program Editor

The program editor is used to edit Natural objects of the following types: program, subprogram, subroutine, help routine, copycode, class and text. The program editor is based on the Software AG Editor, thus it provides many of the same functions. This section describes all of the functions provided by the program editor.

- Editor Basics
  - Invoking the Editor
  - Leaving the Editor
  - Editor Screen
  - Editor Profile
  - Editor Profile Commands
  - Modifying Profile Settings
  - Editor Buffer-Pool Settings
  - Using Cut and Paste
- Commands
  - Scrolling Commands
  - Split-Screen Commands
  - Editor Commands
    - Command Syntax
    - Common Command Options
    - Line Commands

See also Renumbering of Source-Code Line Number References in the Natural Reference Documentation.

---

# Editor Basics

## Invoking the Editor

You invoke the program editor with the EDIT command. This command is described in the section System Commands.

## Leaving the Editor

To leave the editor, you issue one of the following commands: EXIT, CANCEL, or a period (.). Before leaving the editor, you are prompted to enable you to save any changes you have made.

## Editor Screen

When you invoke the program editor, an edit screen similar to the following appears:

```

>> Columns 001 072 << Program PROG01 Lines 45 User SAG
COMMAND ===> Mode Struct Lib SAGLIB
000010 DEFINE DATA LOCAL
000020     1 FEL (A070)
000030     1 CULFEL (A010/1:400)
000040     1 #SCSTR (A20)
    
```

If you specified no object name and the source work area is empty, an empty screen appears. Otherwise a program appears as displayed above. The left-most six columns contain line numbers for the program.

The editor contains the following output fields:

Field Name	Description
Columns	Columns currently displayed. You can enter text beyond the 72nd column, but to ensure that data can be handled by other platforms you are advised to use only Columns 1-72.
Program	Name of the program currently in working storage. If no name was specified when the editor was invoked, then this field is empty.
Lines	Total number of lines currently used by the editor (data lines and information lines).
Mode	Program mode: structured or reporting.
User	ID of the current user.
Lib	Current library name.

## Editor Profile

Each user has an editor profile with parameters which can be set according to individual needs. The first time you invoke the editor, it uses the default values determined by your administrator. There are two ways for you to modify your editor profile settings:

- on a temporary basis at the session level from the command line (example: CAPS OFF/ON). Settings modified in this way are valid for the remainder of the editing session or until you change them again.
- on a permanent basis at the system level from the editor profile utility. Settings modified in this way are valid for each new Natural session. They are activated when you leave Natural and start it again. These settings can be overridden at the session level by temporary settings (see above).

## Displaying Profile Settings

To display the current editor profile settings, issue the PROF command. The following lines appear at the top of the editor screen:

```
>> -----Columns 001 072 << Program           Lines 850   User SAG
COMMAND ==>                               Mode  Struct Lib  PROF
***** ***** top of data *****
=prof> date: 10/26/2001 10:03:12 user: SAG   init size: 844 size: 845
=prof> var   - 250,..recovery off (100 0)...autosave off... empty line off
=prof> mask off.caps off.hex off  nulls on .autoren std off..auto order
=prof> log off .mso off.fix off .escape on + . tabs off
=prof> advance on .protect off.limit on 5000
```

The individual items of the editor profile are described in the following table.

Item	Description	Command
<b>date</b>	The current date and time.	Non-modifiable
<b>user</b>	The current logon user.	Non-modifiable
<b>init size / size</b>	Number of lines in program when editor was invoked. / Current number of lines in the program, excluding information lines (for example profile lines and message lines).	Non-modifiable
<b>var</b>	Specifies current line length.	Non-modifiable
<b>autosave</b>	Activates/deactivates automatic save when EXIT command is issued.	AUTOSAVE
<b>empty line</b>	Specifies if lines containing only space characters are to be deleted automatically.	EMPTY ON/OFF
<b>mask</b>	Activates/deactivates the mask line function.	MASK ON/OFF
<b>caps</b>	Specifies whether data are to be translated into upper case.	CAPS ON/OFF/PGM
<b>hex</b>	Specifies whether data are to be displayed in hexadecimal format.	HEX ON/OFF
<b>autoren</b>	Function not yet available.	
<b>auto order</b>	Automatically justifies text within defined boundaries.	AORDER ON/OFF
<b>log</b>	Enables/disables log file. When enabled, UNDO command can be used to backout last changes.	LOG ON/OFF
<b>mso</b>	Enables/disables multi-session operations such as copy.	MSO ON/OFF
<b>fix</b>	Specifies whether fixed number of columns are displayed and how many columns are to be fixed.	FIX ON/OFF <i>n</i>
<b>tabs</b>	Activates/deactivates tabulation.	TABS ON/OFF
<b>advance</b>	Specifies whether the cursor moves to the next line automatically after a line update.	ADVANCE ON/OFF/PAGE
<b>protect</b>	Specifies protection of line numbers.	PROTECT ON/OFF/INS
<b>limit</b>	Specifies the maximum number of lines to be searched by a FIND or RFIND command.	LIMIT <i>n</i>

## Editor Profile Commands

The following direct commands can be used in the Editor Profile instead of the corresponding PF keys. Direct commands have to be entered in the command line at the bottom of the editor profile screen.

Command	PF Key	Function
<b>CANCEL</b>	<b>PF12</b>	Cancels the current function and redisplay the screen from which it was invoked. Any modifications made to the profile have no effect for the current session.
<b>EXIT</b>	<b>PF3</b>	Invokes the exit function prompt window.
<b>FLIP</b>	<b>None</b>	Toggles between PF 1-12 and PF 13-24.
<b>READ</b>	<b>PF6</b>	Reads the profile parameters for the User ID currently contained in the Profile Name field. Any modifications made so far, but not yet saved, are overwritten (valid only for the Editor Profile main menu).
<b>SAVE</b>	<b>PF5</b>	Saves all currently valid profile parameters both for the current session and on the database. However, it does <b>not</b> leave the current function (valid only for the Editor Profile main menu).

## Modifying Profile Settings

This section describes how to modify your editor profile settings so that the changes take effect not only for the current session (as described in section Displaying Profile Settings), but also for all future edit sessions. To modify your settings, issue the PROFILE command. A screen appears with the following options:

Option	Function
<b>Save</b>	Saves current profile.
<b>Modify</b>	Invokes input screen for modifying editor default settings for PF keys, commands, buffer pool, FIND command parameters.
<b>Read</b>	Reads editor profile of another user which you specify in Profile Name field. The settings from the external profile can then be saved under your profile name.
<b>Tech Info</b>	Displays information for your edit session: library name, program name, object type, Natural mode (report, structured), operating system, steplib (see the section Displaying Technical Information).

## Modifying Editor Default Settings

If you select "Modify" from the main menu, a screen is displayed with the following options:

Option	Function
<b>PA/PF Keys</b>	Modify PF-key assignments.
<b>Commands</b>	Modify command defaults.
<b>FIND Parameters</b>	Modify FIND command parameter settings.
<b>General Defaults</b>	Modify prompt window on leaving editor, number of lines displayed before error line.

## Modifying PF-Key Assignments

If you select "PA/PF Keys" from the main menu, a screen is displayed with the current PA/PF-key assignments. You can modify the assignments at any time during an edit session. These assignments are then valid for all future sessions or until you modify them again. The default PF-key settings for the editor are:

PF Key	Command	Function
PF1	HELP	Invokes Natural online help facility.
PF2	SAVE	Saves program in edit area.
PF3	EXIT	Exits editor.
PF4	RUN	Checks and runs program in edit area.
PF5	RFIND	Repeats last FIND command.
PF6	STOW	Saves and catalogs program in edit area.
PF7	UP, -	Scrolls upwards.
PF8	DOWN, +	Scrolls downwards.
PF9	CHECK	Checks program.
PF10	HOME	Places cursor in command line.
PF11	UNDO	Backs out last change made to editor.
PF12	CANCEL	Exits editor (after prompting you to save changes).
PF13	PROFILE	Invokes editor profile utility.
PF14	RESET	Resets pending line commands/deletes line labels.
PF15	SWAP	Toggles cursor between upper/lower split screens.
PF16	LAST	Performs most recent command.
PF17	RCHANGE	Repeats last CHANGE command.
PF18	FLIP	Toggles PF-key display between 1-12 and 13-24.
PF19	TOP, --	Scrolls to beginning of program.
PF20	BOTTOM, ++	Scrolls to end of program.

To modify a PF-key assignment, simply overtype the name of an existing command (editor command, system command or program name) with another command (maximum five characters).

### Modifying Command Defaults

In this function, you can enable/disable selected editor commands and specify default characters for editor commands. The input fields contained in the screen are explained in the table below. For more information on a specific field, enter a "?" in the field and press ENTER.

Field Name	Description
aorder	Enables/disables autoorder function.
autosave	Enables/disables autosave function.
caps	Specifies whether data are to be translated into upper case.
cols	Specifies whether line is to be displayed showing columns positions.
decimal character	Specifies character used to mark decimal position in numbers used in tabs. (see example TABS DECIMAL).
empty	Specifies if lines containing only space characters are to be deleted automatically.
escape	Specifies whether escape character is used to precede line commands.
escape character	Specifies which character is to be used to precede line commands.
fix	Specifies whether fixed number of columns are displayed beginning with column 1. The number of fixed columns is specified in "fixlength" below.
fixlength	Specifies number of columns starting with column 1 to remain in display when scrolling right.
hex	Specifies whether data are to be displayed in hexadecimal format.
justification	Enables/disables justification or specifies justification type.
limit	Specifies the maximum number of lines to be searched by a FIND or RFIND command.
log file	Enables/disables log file. When enabled, UNDO command can be used to backout last change.
mask line	Enables/disables the mask line function.
message line	Enables/disables message output.
mso	Enables/disables multiple-session operations, such as copies between split-screen sessions.
nulls	Specifies whether the end of each source-code line is to be filled with null characters.
recovery	Function not available.
scroll mode	Specifies how scrolling is to be performed (pagewise, half-pagewise, to cursor)
tabs	Enables/disables tabulation.
tabulator character	Specifies logical tabulation character used to automatically move input to a specific tab position.

## Modifying FIND Command Defaults

If you select "FIND parameters" from the main menu, a screen is displayed with the current parameter settings. These settings are used whenever you issue the FIND command without parameters from the command line of the program editor. You have the opportunity to modify these values in a window before the search is performed.

The following parameters can be modified:

Parameter	Specifies
<b>FIND string</b>	The string to be searched.
<b>Search from/to column</b>	The beginning and end columns of the range of column numbers to be searched.
<b>Search from/to label</b>	The start label and end label of the range of line numbers to be searched.

## Modifying General Defaults

If you select "General Defaults" from the main menu, a screen is displayed with the current defaults. You can modify these settings as required. A short description of each setting follows:

Setting	Specifies
<b>Prompt Window for EXIT Function</b>	Whether the prompt window is to appear when leaving the program editor using the EXIT function.
<b>Prompt Window for CANCEL Function</b>	Whether the prompt window is to appear when leaving the program editor using the CANCEL function.
<b>Lines before error line</b>	The number of lines to be displayed at the top of the page preceding the line containing a syntax error. If you specify zero, then the line containing the syntax error is placed in the first line of the edit area.

## Displaying Technical Information

If you select "Technical Information" from the main menu, a screen is displayed with information about the object currently being edited and the computing environment. The following items are provided:

- user name;
- library name;
- program name;
- object type;
- programming mode;
- operating system;
- steplib.

For further information, see the system command TECH.

## Editor Buffer-Pool Settings

Under certain circumstances it may be necessary to increase the size of the buffer pool above the default setting of 400kb. See the profile parameter description of EDTBPSIZE.

## Multiple Editor Sessions

When you open a new editing session from the program editor, the old session is stored to the buffer pool if the profile parameter EDTRB is set. When you leave the new session with CANCEL or EXIT, the old session is freed from the buffer pool. The NEXT command can be used to jump from one active session to another.

If the EDTRB parameter is switched OFF, the Ring Buffer is not used, and the new session replaces the present session. If the AUTOSAVE parameter is switched on, the contents of the corresponding session are saved to FUSER.

If you intend to use the LOG command, we recommend to increase the size of the buffer pool to 1 MB. Changes are generally stored in the buffer-pool editor.

## Using Cut and Paste

Before pasting text to the program editor, the following commands must be executed:

1. Execute PROTECT ON to protect the prefix area.
2. Execute ESCAPE ON '.' to enable the escape character.
3. Execute .I(n) in the data area to create some empty lines.

4. Now paste the text to the program editor.
5. Use line commands '(n' (Shift left) to delete some copied line numbers

**Note:**

To copy some text from one program to another, you can also use the Split-Screen command. Mark your lines with the line commands 'CC .. CC' in the SPLIT session and copy it with the line command 'A' to the edit session. After doing this, execute 'SPLIT OFF'.

## Commands

The program editor provides two types of commands for editing:

- Editor commands
- Line commands

### Note:

Depending on the configuration of your installation, editor commands and line commands may be entered in lower case. In this section, however, all commands are shown in upper case to distinguish them as commands.

### Scrolling Commands

Command	Function
<b>BOTTOM</b> or ++	Scrolls to the end of the object being edited.
<b>TOP</b> or --	Scrolls to the beginning of the object being edited.
<b>DOWN</b>	Scrolls forwards by the amount specified by the scroll mode.
<b>DOWN</b> <i>n</i>	Scrolls forwards by <i>n</i> lines.
+ <i>n</i>	Scrolls forwards by <i>n</i> lines.
<b>UP</b>	Scrolls backwards by the amount specified by the scroll mode.
<b>UP</b> <i>n</i>	Scrolls backwards by <i>n</i> lines.
- <i>n</i>	Scrolls backwards by <i>n</i> lines.
<b>LEFT</b>	Scrolls to the left by the amount specified by the scroll mode.
<b>LEFT</b> <i>n</i>	Scrolls to the left by <i>n</i> columns.
<b>RIGHT</b>	Scrolls to the right by the amount specified by the scroll mode.
<b>RIGHT</b> <i>n</i>	Scrolls to the right by <i>n</i> columns.
<b>FIX</b> <i>n</i>	Specifies the number of columns <i>n</i> , starting with column 1, to remain in display when scrolling to the right.

### Split-Screen Commands

In split-screen mode, you can simultaneously edit one Natural object while viewing another. The following items can be viewed in split-screen mode:

- views;
- Natural objects, such as data areas, programs, subprograms, subroutines, help routines, copycodes, texts, maps, classes.

Note that it is not possible to edit the object in the display (bottom) section. It is, however, possible to mark lines in this section and copy them into the edit section (using the line commands C and CC).

The following figure shows the program editor in split-screen mode with the source code of a program in the editing section (upper half) and a local data area in the display section (lower half):

```

>> -----Columns 001 072 << Program NEIPBNXT Lines 187 User SAG
Command ==>> Lib SYSLIB
001720 IF (RC NE 0)
001730 then
001740 reset CMD_LINE_2
001750 IF (s_prog ne ' ')
001760 then
001770 assign CMD_LINE_PGM = S_PROG
001780 assign CMD_LINE_LIB = S_LIB
001790 END-IF
001800 END-IF
>> -----Columns 001 072 << Local NPFLCONS Lines 158 User SAG
Command ==>> Lib SYSLIB
***** ***** top of data *****
000001 DEFINE DATA LOCAL
000002 1 NPF_F_READ(A1)
000003 CONST
000004 <'R'>
000005 1 NPF_F_WRITE(A1)
000006 CONST
000007 <'W'>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Help Save Exit Run Rfind Stow - + Check Home Undo Canc
    
```

Note that, because it is not possible to make data modifications to the display section, not all editor commands are available. PF keys are reserved for the edit section at the top of the screen, thus commands meant for the display section must be issued using the command line.

**Note:**

The SWAP command can be used to move the cursor swiftly between the command lines of the display and edit sessions.

The following commands can be used to display and position an object in split-screen mode. All commands begin with an "S" or with SPLIT to indicate the working mode.

Command	Function
<b>SPLIT DATA</b> <i>name [library]</i>	Display data area (global, local, parameter).
<b>SPLIT PROGRAM</b> <i>name [lib]</i>	Display program, subprogram, subroutine, help routine, copy code, text, map.
<b>SPLIT VIEW</b> <i>name [SHORT]</i>	Display view (DDM, as defined in Predict or SYSDDM). If SHORT is specified, the view is listed in short form (that is, only the Adabas short names and corresponding Natural field names are displayed) without any field header or field edit mask information.
<b>SPLIT END</b>	Terminate split-screen mode.

With DATA, PROGRAM and VIEW, an asterisk "\*" can be used for *name* to display a list of all available objects. If the "\*" is preceded by one or more characters, only those objects whose names begin with these characters are displayed.

## Editor Commands

Editor commands are issued from the command line. Some frequently used commands can be issued using PF keys.

### Command Syntax

<b>CURSOR</b>	Terms in upper-case bold letters are used for commands.
<u>  </u>	Full underlining indicates a default value; partial underlining indicates an abbreviated form.
<i>string</i>	Terms in lower-case italic letters are used for parts of syntax or commands whose values have to be supplied by the user.
[ ]	Elements contained within square brackets are optional.  If the square brackets contain several lines stacked one above the other, each line is an optional alternative. You may choose at most one of the alternatives.
{ }	If the braces contain several lines stacked one above the other, each line is an alternative. You must choose exactly one of the alternatives.
<b>...4</b>	Terms preceding the ellipsis may be repeated. A numeric constant after an ellipsis limits the number of times the term may be repeated.

### Common Command Options

There are some options which are available with several editor commands. These options are described below for all commands in which they can be specified.

### Line Specifications

With these options, you can restrict the effect of a command to a certain range of lines:

<b>.X</b>	The command affects only the lines from the line labelled ".X" to the last line.
<b>.X.Y</b>	The command affects only the lines from the line labelled ".X" to the line labelled ".Y".

X and Y can also be any label of 1 to 4 alphabetical characters (see LABEL command).

### Column Specifications

With these options, you can restrict the effect of a command to a certain range of columns. These column numbers refer to the actual source-code columns; the line numbers preceding the source code are not counted. So, if you specify column 1 with a command, this may physically be the 8th column of your screen, but it is in fact the 1st column of the source code you are editing.

<i>n</i>	The command affects only lines in which the specified string begins in column n (that is, the first character of the string must be in column n).
<i>n</i> <i>m</i>	The command affects only lines in which the specified string occurs anywhere between columns n and m.

### Displayed or Non-displayed Lines

With one of the following options, you can specify that only excluded or only included lines are to be affected by a command:

<b>NX</b>	The command affects only non-excluded lines; that is, lines which are currently being displayed.
<b>X</b>	The command affects only excluded lines; that is, lines which are currently <i>not</i> being displayed as specified by the EXCLUDE command. An excluded line remains excluded from display if an editor command function is performed on it.

### Direction of Operation

With these options, you can specify the direction in which a command is to operate:

<b>NEXT</b>	The command affects the next line (starting from the cursor position) in which the specified <i>string</i> occurs.
<b>PREV</b>	The command affects the line that contains the previous occurrence of the specified <i>string</i> .
<b>FIRST</b>	The command affects the first line in which the specified <i>string</i> occurs.
<b>LAST</b>	The command affects the last line in which the specified <i>string</i> occurs.
<b>ALL</b>	The command affects all lines in which the specified <i>string</i> occurs.

### Special Occurrences

With these options, you can specify whether only special occurrences of the specified *string* are to be affected by a command:

<b>CHARS</b>	The command affects any line in which the specified <i>string</i> occurs.
<b>WORD</b>	The command affects only those lines in which the specified <i>string</i> forms a word.
<b>PREFIX</b>	The command affects only those lines in which the specified <i>string</i> is the beginning of a word.
<b>SUFFIX</b>	The command affects only those lines in which the specified <i>string</i> is the end of a word.

### ADVANCE

<b>ADVANCE</b> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center; padding: 2px;"><u>ON</u></td> </tr> <tr> <td style="text-align: center; padding: 2px;">OFF</td> </tr> <tr> <td style="text-align: center; padding: 2px;">PAGE</td> </tr> </table>	<u>ON</u>	OFF	PAGE
<u>ON</u>			
OFF			
PAGE			

This command is used to specify whether the cursor moves to the next line automatically after a line update.

<b>ON</b>	The cursor moves to the next line after an update.
<b>OFF</b>	The cursor does not move to the next line after an update.
<b>PAGE</b>	The line containing the cursor is placed at the top of the edit area after an update.

If an unqualified ADVANCE command is issued, it is interpreted as ADVANCE ON.

### AORDER

<b>AORDER</b> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center; padding: 2px;"><u>ON</u></td> </tr> <tr> <td style="text-align: center; padding: 2px;">OFF</td> </tr> </table>	<u>ON</u>	OFF
<u>ON</u>		
OFF		

This command is used to specify whether newly-entered text is to be automatically justified within the set boundaries.

If an unqualified AORDER command is issued, it is interpreted as AORDER ON. The base setting can be changed by editing your profile.

### AUTOSAVE

$\left\{ \begin{array}{l} \text{AUTOSAVE} \\ \text{ASAVE} \end{array} \right\} \left[ \begin{array}{l} \text{ON} \\ \text{OFF} \end{array} \right]$
---

This command is used to specify whether the editor executes an automatic SAVE command when you issue the EXIT command.

If an unqualified AUTOSAVE command is issued, it is interpreted as an AUTOSAVE ON command.

### BNDS

$\text{BNDS} \left[ \begin{array}{l} n \ m \\ n \end{array} \right]$
--

This command is used to restrict the effect of certain commands to a specific range of columns.

These boundaries apply to the editor commands FIND, CHANGE, CENTER, ORDER, JLEFT and JRIGHT, and their corresponding line commands (TC, TO, LJ, RJ, etc.).

<i>n</i>	The number of the column at which the left boundary is to be placed.
<i>m</i>	The number of the column at which the right boundary is to be placed.

If *n* and *m* are omitted, the boundaries are set at the first and last column of the edit area.

To see the current boundary settings, issue the BNDS line command.

### BOTTOM

<u>B</u> OTTOM
----------------

This command is used to scroll to the end of the object being edited.

### CANCEL

<u>C</u> ANCEL
----------------

This command cancels all changes made after the last SAVE or STOW command and leaves the editor.

### CAPS

```
CAPS [ ON ]
      [ OFF ]
      [ PGM ]
```

This command is used to switch upper-case translation on and off. The command applies only to new or modified lines.

<b>ON</b>	Line is translated to upper case.
<b>OFF</b>	Line is not translated; that is, it remains as entered.
<b>PGM</b>	Line is translated to upper case (except for comments, which remain as entered).

The CAPS command issued without a parameter has the same effect as CAPS ON.

### CENTER

```
CENTER { ALL }
        { n }
        { n m }
```

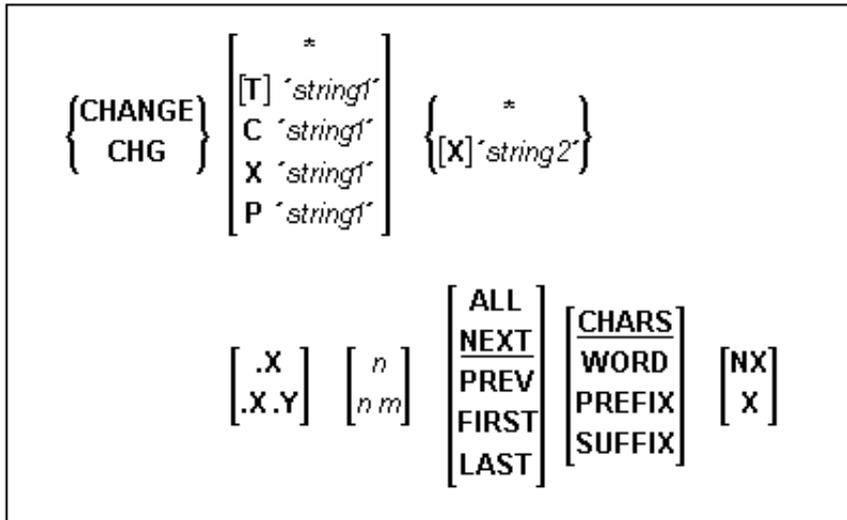
This command is used to center source code.

<b>ALL</b>	Centers the source code of all lines.
<i>n</i>	Centers the source code from line <i>n</i> to the last line.
<i>n m</i>	Centers the source code from line <i>n</i> to line <i>m</i> .

The CENTER command applies only within the horizontal boundaries as set with the editor command BNDS.

For centering, you can also use the line commands TC and TCC.

### CHANGE



This command is used to replace a character string (*string1*) by another character string (*string2*).

You can specify the string to be replaced (*string1*) as follows:

<b>T'</b> <i>string1</i> '	Delete lines that contain the <i>string</i> irrespective of lower case or upper case. This is the default.
' <i>string1</i> '	Same as <b>T'</b> <i>string</i> '.
<b>C'</b> <i>string1</i> '	Delete lines that contain the <i>string</i> exactly as specified.
<b>X'</b> <i>string1</i> '	Delete lines that contain the string which corresponds to the specified hexadecimal character <i>string</i> .
<b>P'</b> <i>string1</i> '	Delete lines that contain the <i>string</i> which includes the following wildcard characters: = any character § alphabetical character # numeric character \$ special character ^ non-blank character - non-numeric character < lower-case letter > upper-case letter
*	Use the replacement string specified in a previous command (for example, FIND, CHANGE, EXCLUDE).

If you want an apostrophe to be part of *string1* or *string2*, you must write it as two apostrophes.

All other options of the CHANGE command are described in the section Common Command Options.

### Using CHANGE Together with Other Commands

To repeat the execution of a CHANGE command, you use the command RCHANGE.

To search the entire source code for a character string and then decide occurrence by occurrence whether to replace it by another character string, you can use a combination of the commands FIND, CHANGE, RFIND and RCHANGE:

First, you search for the string:

**FIND** *'string'*

When the string has been found, you can decide whether to:

- replace it:  
**CHANGE** *'string'* *'new-string'*
- or search for the next occurrence of the string by repeating the FIND command:  
**RFIND**

When the next occurrence of the string has been found, you can again decide whether to:

- replace it by repeating the CHANGE command:  
**RCHANGE** *'*
- or search for the next occurrence of the string by repeating the FIND command:  
**RFIND**

**Examples of the CHANGE Command****Example 1:**

**CHG 'LOW' 'HIGH'**

This command replaces the first occurrence of "low" by "high" (regardless of upper or lower case).

**Example 2:**

**CHG C'OPS' 'SPF' .X .Y 28 32 ALL**

This command changes "OPS" (exactly as entered here) into "SPF"; it changes all occurrences in the block of lines labelled by ".X" and ".Y" and between columns 28 and 32.

**Example 3:**

**CHG C'NAME' 'APPL' .X .Y ALL PREFIX NX**

This command changes all occurrences of prefix "NAME" (exactly as entered here) into "APPL" in all displayed lines in the block labelled by ".X" and ".Y".

**Example 4:**

**CHG \* 'NEW'**

This command replaces the next occurrence of the string specified in the last CHANGE command by the string "NEW".

**Example 5:**

**CHG 'OLD' \***

This command replaces the next occurrence of the string "OLD" by the same new string as specified in the last CHANGE command.

**COLS**

COLS [ON OFF]
------------------

This command displays a line at the top of the edit area showing column positions.

To display the column positions, you can also use the line commands COLS.

### COPY

```
COPY { *object-name } [ssss [nnnn] ]
```

This command copies an external object to the editor and inserts the object after the line marked with the A line command or before the line marked with the B line command (see line commands). The optional parameter *ssss* can be used to specify the line of the external object at which the include operation is to begin. The optional parameter *nnn* can be used to specify the number of lines to be copied.

### CURSOR

```
CURSOR
```

This command returns the cursor to the command field when you next press ENTER.

### CWINDOW

```
CWINDOW [ n ]  
[ n m ]
```

This command is used to copy a data window according to the command parameters.

<i>n</i>	The number of the line in which the data window is to be inserted.
<i>m</i>	The number of the column in which the data window is to be inserted.

### DELETE

```
DELETE [ * ]  
[ [T] 'string' ]  
[ 'string' ]  
C 'string'  
X 'string'  
P 'string' ]  
[ .X ] [ n ]  
[ .X.Y ] [ n m ]  
[ ALL ]  
[ NEXT ]  
[ PREV ]  
[ FIRST ]  
[ LAST ]  
[ CHARS ]  
[ WORD ]  
[ PREFIX ]  
[ SUFFIX ]  
[ NX ]  
[ X ]
```

This command is used to delete lines.

You can specify that only lines which contain a specified character *string* are to be deleted. You have the following options:

<b>T'</b> <i>string</i>	Delete lines that contain the <i>string</i> irrespective of lower case or upper case. This is the default.
<b>'</b> <i>string</i>	Same as <b>T'</b> <i>string</i> .
<b>C'</b> <i>string</i>	Delete lines that contain the <i>string</i> exactly as specified.
<b>X'</b> <i>string</i>	Delete lines that contain the string which corresponds to the specified hexadecimal character <i>string</i> .
<b>P'</b> <i>string</i>	Delete lines that contain the <i>string</i> which includes the following wildcard characters: = any character § alphabetical character # numeric character \$ special character ^ non-blank character - non-numeric character < lower-case letter > upper-case letter
*	Use the search string specified in a previous command (for example, FIND, CHANGE, EXCLUDE).

All other options of the DELETE command are described in the section Common Command Options.

If you enter the DELETE command without any parameters, the current line is deleted.

**Example 1:**

**DEL C'NAME' 1 20 ALL PREFIX NX**

This command deletes all lines that contain the string "NAME" (in upper case exactly as entered here) as a prefix to a word in all lines not excluded from display, if "NAME" occurs between columns 1 and 20.

**Example 2:**

**DEL C'Abc' .X.Y 10 30 ALL**

This command deletes all lines that contain the string "Abc" (exactly as entered here) between columns 10 and 30 within the block of lines labelled by ".X" and ".Y"

To delete lines, you can also use the line commands D, Dn and DD.

**DOWN**

<b>DOWN</b> [n]
-----------------

This command is used to scroll downwards in the source code.

The parameter *n* specifies the number of lines to be scrolled downwards. If *n* is omitted, the scroll amount is determined by the scroll mode.

**DWINDOW**

<b>DWINDOW</b>
----------------

This command is used to delete the last defined data window.

**DX, DY, DX-Y**

<b>DX</b>
<b>DY</b>
<b>DX-Y</b>

These commands are used to delete marked lines in the program editor.

- The DX command deletes the line marked with the .X label.
- The DY command deletes the line marked with the .Y label.
- The DX-Y command deletes all lines between the .X and .Y labels.

**EMPTY**

<b>EMPTY</b> [ <b>ON</b> ]
[ <b>OFF</b> ]

This command controls the deletion of blank lines in the editor.

<b>OFF</b>	Empty lines are not deleted.
<b>ON</b>	Empty lines are deleted.

If you enter EMPTY without any parameter, it is interpreted as EMPTY OFF.

**ESCAPE**

<b>ESCAPE</b> [ <b>ON</b> ]	[ <i>character</i> ]
[ <b>OFF</b> ]	

This command activates/deactivates the escape character to precede line commands entered in the first column of the source code.

The parameter *character* is the special character to be used. The default escape character is the period (.).

If you issue the ESCAPE command without any parameter, it is interpreted as ESCAPE ON.

**EX, EY, EX-Y**

EX  
EY  
EX-Y

These commands are used to delete lines in a program.

- The EX command deletes *all lines* preceding the line marked with the .X label.
- The EY command deletes *all lines* following the line marked with the .Y label.
- The EX-Y command deletes *all lines* preceding the .X label and following the .Y label.

**EXCLUDE**

{EXCLUDE}     $\left[ \begin{array}{l} * \\ \text{[T]} \text{'string' } \\ \text{'string' } \\ \text{C 'string' } \\ \text{X 'string' } \\ \text{P 'string' } \end{array} \right]$      $\left[ \begin{array}{l} \text{.X} \\ \text{.X.Y} \end{array} \right]$      $\left[ \begin{array}{l} n \\ n\ m \end{array} \right]$      $\left[ \begin{array}{l} \text{ALL} \\ \text{NEXT} \\ \text{PREV} \\ \text{FIRST} \\ \text{LAST} \end{array} \right]$      $\left[ \begin{array}{l} \text{CHARS} \\ \text{WORD} \\ \text{PREFIX} \\ \text{SUFFIX} \end{array} \right]$

This command is used to exclude lines from being displayed.

You can specify that only lines which contain a specified character *string* are to be excluded from display. You have the following options:

<b>T</b> 'string'	Delete lines that contain the <i>string</i> irrespective of lower case or upper case. This is the default.
'string'	Same as <b>T</b> 'string'.
<b>C</b> 'string'	Delete lines that contain the <i>string</i> exactly as specified.
<b>X</b> 'string'	Delete lines that contain the string which corresponds to the specified hexadecimal character <i>string</i> .
<b>P</b> 'string'	Delete lines that contain the <i>string</i> which includes the following wildcard characters: = - any character § - alphabetical character # - numeric character \$ - special character ^ - non-blank character - - non-numeric character < - lower-case letter > - upper-case letter
*	Use the search string specified in a previous command (for example, FIND, CHANGE, EXCLUDE).

All other options of the EXCLUDE command are described in the section Common Command Options.

If you enter the EXCLUDE command without any parameters, the current line is excluded from display.

**Example 1:**

**EXCLUDE .X .Y**

This command excludes lines from the line labeled .X to the line labeled Y.

**Example 2:**

**EXCLUDE C'NAME' ALL PREFIX**

This command excludes from display all lines which contain "NAME" (in upper case as entered here) as a prefix to a word.

To re-display excluded lines, you use the editor command INCLUDE.

**EXIT**

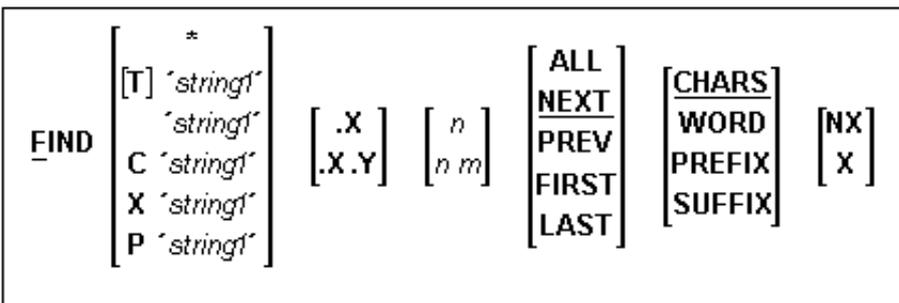


This command is used to leave the editor. If any changes have been made since the last SAVE or STOW, then you are prompted to save your changes or exit without saving.

**Note:**

If AUTOSAVE is set to ON, then you are not prompted before exiting the session; your changes are saved automatically.

**FIND**



This command is used to search for a specific character *string*. The cursor is placed on the beginning of the first found *string*. If the line containing the *string* was excluded from display, it is displayed when found.

You can specify the *string* as follows:

<b>T'</b> <i>string</i>	Search for the <i>string</i> irrespective of lower case or upper case. This is the default.
<b>'</b> <i>string</i>	Same as <b>T'</b> <i>string</i> .
<b>C'</b> <i>string</i>	Search for the <i>string</i> exactly as specified.
<b>X'</b> <i>string</i>	Search for the string that corresponds to the specified hexadecimal character <i>string</i> .
<b>P'</b> <i>string</i>	Search for a <i>string</i> which includes the following wildcard characters: = - any character § - alphabetical character # - numeric character \$ - special character ^ - non-blank character - - non-numeric character < - lower-case letter > - upper-case letter
<b>*</b>	Search for the <i>string</i> specified in the previous FIND command.

If you want an apostrophe to be part of the *string*, you must write it as two apostrophes.

All other options of the FIND command are described in the section Common Command Options.

**Example 1:**

**F C'NAME' .X .Y ALL PREFIX X**

This command searches for any occurrence of "NAME" exactly as entered here as a prefix of a word in any excluded line within the block delineated by ".X" and ".Y".

**Example 2:**

**F C'HILITE' X PREV**

This command searches for the previous occurrence of "HILITE" exactly as entered here in any excluded line.

**Example 3:**

**F P'RCV#' .X .Z 20 30**

This command searches for any 4-character string that begins with "RCV" and whose fourth character is numeric. It searches within the block of lines delineated by ".X" and ".Z" and between columns 20 to 30.

**Example 4:**

**F X'6C' SUFFIX NX**

This command searches for the character that is hexadecimally represented as "6C". Only those occurrences of the character that are at the end of word are found. The search is valid for non-excluded lines only.

**Example 5:**

**F '''w'**

This command searches for the following character string: 'w

**Example 6:**

**F 'r''w'**

This command searches for the following character string: **r'w**

**Example 7:****F '''**

This command searches for an apostrophe.

The FIND command differs from the LOCATE command in the following ways:

- The FIND command is more effective for text searches while the LOCATE command is used primarily to find line numbers or line labels.
- The LOCATE command finds only text in upper case beginning in column one of the editor. In addition, in order to find a string, the data in the editor must be in alphabetical order.
- When a line is located with the LOCATE command, the cursor is placed in the prefix area and the line is placed at the top of the editor; with the FIND command, the cursor is placed on the string searched and the line is not necessarily placed at the top of the editor.

To repeat the execution of a FIND command, use the command RFIND.

**FIX**

<b>FIX</b> [n]
----------------

This command is used to specify the number of columns *n*, starting with column 1, to remain in display when scrolling to the right.

**FLIP**

<b>FLIP</b>
-------------

This command is used to toggle the PF key display between PF1-12 and PF13-24.

**HEX**

<b>HEX</b> [ON] [OFF]
--------------------------

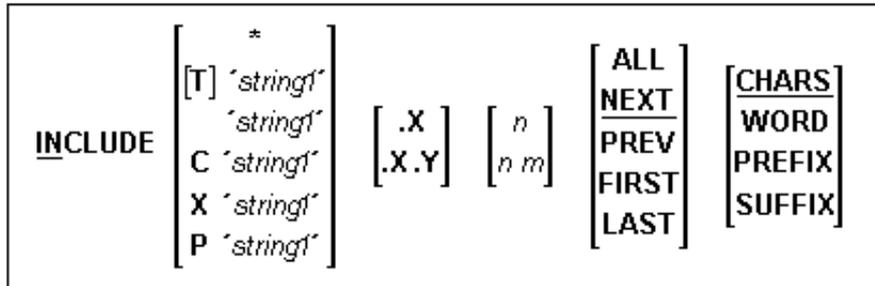
This command is used to switch hexadecimal display mode on or off.

**HOME**

<b>HOME</b>
-------------

This command returns the cursor to the command field at the next ENTER.

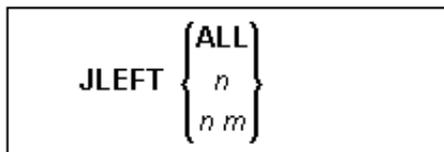
**INCLUDE**



This command is used to re-display lines that were excluded from display by an EXCLUDE command. The command takes the same parameters as the EXCLUDE command.

If you enter the INCLUDE command without any parameters, it includes the first line of an excluded block.

**JLEFT**



This command is used to align source code left-justified.

<b>ALL</b>	Aligns the source code of all lines.
<i>n</i>	Aligns the source code from line <i>n</i> to the last line.
<i>n m</i>	Aligns the source code from line <i>n</i> to line <i>m</i> .

The JLEFT command applies only within the horizontal boundaries as set with the editor command BNDS.

**Example:**

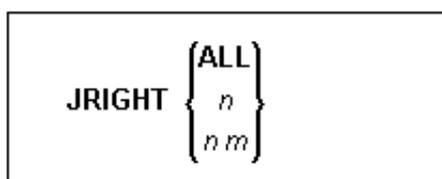
**BNDS 10;JLEFT 15 20**

The source code between column 10 and the rightmost column of your screen in lines 15 to 20 is left-aligned to column 10.

For left-justification, you can also use the line commands LJ and LJJ.

See also the editor command JRIGHT.

**JRIGHT**



This command is used to align source code right-justified.

<b>ALL</b>	Aligns the source code of all lines.
<i>n</i>	Aligns the source code from line <i>n</i> to the last line.
<i>n m</i>	Aligns the source code from line <i>n</i> to line <i>m</i> .

The JRIGHT command applies only within the horizontal boundaries as set with the editor command BNDS.

**Example 1:**

**BNDS 4 40;JRIGHT 6 18**

The source code between columns 4 to 40 in lines 6 to 18 is right-aligned to column 40.

**Example 2:**

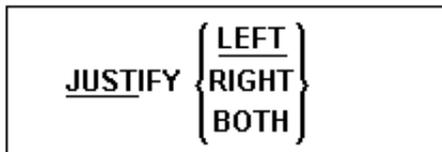
**BNDS 10;JRIGHT 15**

The source code to the right of column 10 from line 15 to the last line is right-aligned to the rightmost column of your editing screen.

For right-justification, you can also use the line commands RJ and RJJ.

See also the editor command JLEFT.

**JUSTIFY**



This command is used to set the justification mode for the line commands TO and TOO.

TO and TOO are used to join source-code lines with subsequent lines. Both commands apply only within the horizontal boundaries as set with the editor command BNDS.

<b>LEFT</b>	The source code is aligned to the left boundary.
<b>RIGHT</b>	The source code is aligned to the right boundary.
<b>BOTH</b>	The source code is aligned to both boundaries.

**Example:**

With these commands, you set the horizontal boundaries to columns 10 and 60, and activate left-justification:

**BNDS 10 60;JUSTIFY LEFT**

When you then mark a line with a TO line command (or a block of lines with two TOO line commands), the source code between columns 10 and 60 in the marked line(s) is left-aligned to column 10.

## LABEL

```
LABEL .label
```

This command is used to mark the current line (that is, the line which is currently at the top of the edit area) with the specified *.label*.

The *label* is a string of 1 to 5 alphanumerical characters.

### Example:

To label the current line with ".X", you enter the command:

## LABEL .X

You can also mark a block of lines with two labels. For example, to mark a block with labels ".X" and ".Y", you first mark the current line (assuming it is the first line of the block to be marked) with ".X" as shown in the example above; then you scroll until the last line of the block is the current line; then you issue the command "LABEL .Y" to mark that line with ".Y".

To mark a line with a label, you can also use the line command ".label".

## LAST

```
LAST
```

This command recalls the editor command last issued. This command is placed in the command line and can be modified as required.

## LC

```
LC [ *
    [T] 'stringf'
      'stringf'
    C 'stringf'
    X 'stringf'
    P 'stringf' ] [ .X ] [ n ] [ ALL
                  [ .X.Y ] [ n m ] [ NEXT
                  [ CHARS ] [ WORD
                  [ X ] [ X ] [ PREFIX
                  [ SUFFIX ]
```

This command is used to change one or more lines to lower case.

You can specify that only lines which contain a specified character *string* are to be changed to lower case. You have the following options:

<b>T'</b> <i>string</i>	Change lines which contain the <i>string</i> irrespective of lower case or upper case. This is the default.
<b>'</b> <i>string</i>	Same as <b>T'</b> <i>string</i> .
<b>C'</b> <i>string</i>	Change lines which contain the <i>string</i> exactly as specified.
<b>X'</b> <i>string</i>	Change lines which contain the string that corresponds to the specified hexadecimal character <i>string</i> .
<b>P'</b> <i>string</i>	Change lines which contains a <i>string</i> that includes the following wildcard characters: = any character § alphabetical character # numeric character \$ special character ^ non-blank character - non-numeric character < lower-case letter > upper-case letter
*	Change lines which contain the <i>string</i> used in the previous command in which a string was specified.

If you want an apostrophe to be part of the *string* , you must write it as two apostrophes.

All other options of the LC command are described in the section Common Command Options.

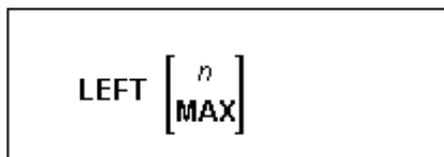
If you enter the LC command without any parameters, the current line is changed to lower case.

**Example:**

**LC C'NAME' .X .Y ALL PREFIX NX**

This command changes to lower case all displayed lines within the block labelled by ".X" and ".Y" if they contain the string "NAME" (in upper case as entered here) as prefix to a word.

**LEFT**

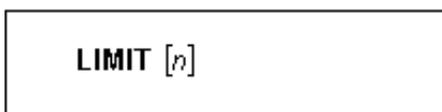


This command scrolls the source code to the left

<i>n</i>	Scrolls <i>n</i> number of columns to the left.
<b>MAX</b>	Scrolls the maximum amount to the left.

If *n* or **MAX** is omitted, the scrolling amount is determined by the scroll mode.

**LIMIT**



With this command, you specify the maximum number of lines to be searched with a FIND or RFIND command. The parameter *n* is the number of lines to be searched.

## LOCATE

[LOCATE]	{ 0 n .label }
----------	----------------------

This command is used to scroll a specific line to the top of the edit area (that is, make it the current line).

The command provides the following options:

0	Makes the first line of the source code current.
<i>n</i>	Makes line <i>n</i> current.
<i>.label</i>	Makes the line labelled <i>.label</i> current.

### Examples:

**LOC 32** Places line number 32 at the top of the edit area.

**32** Same as above.

**LOC .X** Places the line labelled ".X" at the top of the edit area.

## LOG

LOG	[ON OFF]
-----	-------------

This command activates or deactivates the log file.

The log file is a history of all modifications made in the editor since session begin. When the log file is active, each time you press ENTER, the changes made since the previous ENTER are recorded in the log file. Using the UNDO command you can consecutively back out changes made since the beginning of the edit session.

## MASK

MASK	[ON OFF]
------	-------------

This command activates or deactivates the mask function. When the mask function is active, each time you insert a line in the editor, a predefined line of text is entered instead of a blank line. The mask line is defined using the MASK line command, described in the following paragraph. The mask function is useful when you must write several lines of code which are identical or very similar.

To define a mask line, type "mask" over any line number in the editor and press ENTER. An empty line appears in which you can type your mask. This mask is active until you update the mask with a new mask line or until you deactivate the mask function.

When the mask function is activated using MASK ON, the mask line appears in all lines added with a line insertion operation. Note, however, that any inserted line is deleted at the next press of ENTER if nothing is added to it.

The command MASK OFF deactivates the mask function but does not delete the contents of the mask line.

### MSO

```
MSO [ON  
OFF]
```

This command is used to specify whether multiple session operations are possible or not. A multiple session operation is an operation in which data are exchanged between two editing sessions, for example in copying text from one program to another in split-screen mode.

### MWINDOW

```
MWINDOW [ n  
n m]
```

This command is used to move a data window according to the command parameters.

<i>n</i>	The number of the line in which the data window is to be inserted.
<i>m</i>	The number of the column in which the data window is to be inserted.

### NEXT

```
NEXT [ *  
object-name]
```

This command is used to display the *next* parallel editing session, assuming two or more editing sessions are running concurrently. The following command parameters are optional:

*	Displays a list of all concurrently running sessions for selection.
<i>object-name</i>	Calls directly by name a concurrently running editing session.

### NULLS

NULLS  $\left[ \begin{array}{c} \text{ON} \\ \text{OFF} \end{array} \right]$

This command is used to determine if the source-code lines are to be filled with null characters.

<b>ON</b>	The end of each line is filled with null characters.
<b>OFF</b>	Lines are not filled with null characters.

**ORDER**

ORDER  $\left\{ \begin{array}{c} \text{ALL} \\ n \\ n\ m \end{array} \right\}$

This command is used to join source-code lines.

<b>ALL</b>	Joins all lines.
<i>n</i>	Joins the lines from line <i>n</i> to the last line.
<i>n m</i>	Joins lines from line <i>n</i> to line <i>m</i> .

The ORDER command applies only within the horizontal boundaries as set with the editor command BNDS.

Within the set boundaries, the lines are concatenated and are filled to the greatest possible extent; words that do not fit into one line are automatically placed in the next line.

To join source-code lines, you can also use the line commands TF, TO and TOO.

**OWINDOW**

OWINDOW  $\left[ \begin{array}{c} n \\ n\ m \end{array} \right]$

This command is used to overlay a data window according to the command parameters. The moved/copied lines are merged with this window, that is, blank characters in the window are overlaid.

<i>n</i>	The number of the line in which the data window is to be inserted.
<i>m</i>	The number of the column in which the data window is to be inserted.

**POINT**

POINT

This command places the line marked with line command NZ at the top of the data area.

**POWER**

```
POWER
```

This command switches the editor to text-entry mode. You are presented with a blank screen into which you can enter one or more lines of text. After entry, press ENTER and the text is inserted into the first line of the edit area.

**PROF**

```
PROF [n]
```

This command displays your editor profile at the top of the edit screen.

With *n* you specify additional lines to be displayed. Possible values for *n* are:

6	Displays your editor profile and all tab positions (as specified by TABS command).
7	Displays same as 6, plus the mask line (as specified by the MASK command).
8	Displays same as 7, plus boundaries (as specified by the BNDS command).
9	Displays same as 8, plus column numbers (as specified by the COLS command).

**PROFILE**

```
PROFILE
```

This command invokes your editor profile utility. It enables you to modify your editor defaults for current and future sessions. The editor profile utility is described in more detail in section Modify Profile Settings.

**PROTECT**

```
PROTECT [INS]
          [ON]
          [OFF]
```

This command is used to protect the prefix area (line numbers). To enter line commands with the prefix area protected, type the line command in column 1 of the edit area preceded by the escape character.

<b>INS</b>	Protects the prefix area of lines added using the insert line command.
<b>ON</b>	Activates protection.
<b>OFF</b>	Deactivates protection.

**RCHANGE**

**RCHANGE**

This command repeats the last CHANGE command.

**RESET**

**RESET**

This command resets all pending line commands and deletes all line labels.

**RFIND**

**RFIND**

This command repeats the last FIND command.

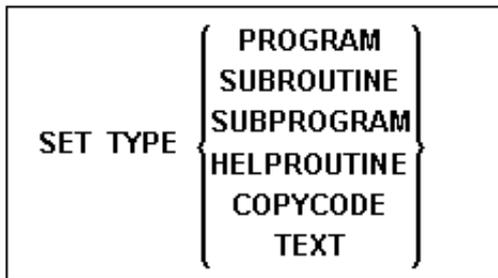
**RIGHT**

**RIGHT**  $\left[ \begin{array}{c} n \\ \text{MAX} \end{array} \right]$

<i>n</i>	Scrolls <i>n</i> number of columns to the right.
<b>MAX</b>	Scrolls the maximum amount to the right.

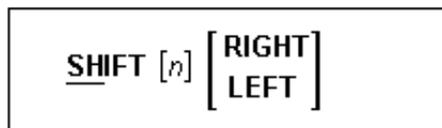
If *n* or MAX is omitted, the scrolling amount is determined by the scroll mode.

**SET TYPE**



This command changes the type of the object currently in the editor work area.

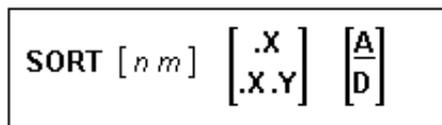
**SHIFT**



This command shifts a block of lines between the .X and .Y labels to the right or left by *n* columns (or until last non-blank character). The default shift is five columns to the right.

<i>n</i>	The number of columns the lines are to be shifted (default value:5).
<b>RIGHT</b>	Shifts block of lines to the right (default).
<b>LEFT</b>	Shifts block of lines to the left.

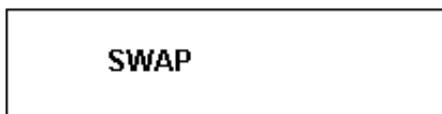
**SORT**



The SORT command sorts lines in the editor in ascending or descending alphabetical order. An unqualified SORT command sorts all data in the object in ascending order.

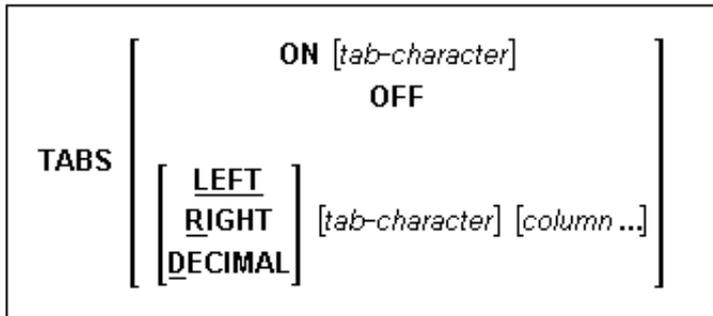
<i>n m</i>	Sorts from column <i>n</i> to column <i>m</i> .
.X	Sorts from line labelled .X to end of object.
.X.Y	Sorts from line labelled .X to line labelled .Y (where .X and .Y are any string of up to four characters).
<b>A</b>	Sorts data in ascending order (A to Z).
<b>D</b>	Sorts data in descending order (Z to A).

**SWAP**



The SWAP command toggles between two objects in split-screen mode. During this operation, the cursor switches from one object to the other.

## TABS



This command is used to control tabulator settings.

You can enable or disable logical or physical tabulation using the command TABS ON/OFF. Tabulation is also enabled by any command that changes a tabulation setting.

For example, the following command enables logical tabulation with the ampersand sign (&) as logical tabulation character:

### TABS &

You set tab positions using the TABS command. For example, the following command sets tabs in columns 10, 20 and 30:

### TABS 10 20 30

You can enter data and automatically move them to a specific tab position by preceding them with a logical tabulation character. One tabulation character moves the data to the next tab position, two tabulation characters move the data to the second tab position, etc.

To display the current TABS command settings, issue the editor command PROF.

To display the current tab positions, issue the line command TABS.

Apart from tab positions, you can specify the following parameters with the TABS command:

<b><u>L</u>LEFT</b>	Places the data left-justified at the tab position.
<b><u>R</u>IGHT</b>	Places the data right-justified at the tab position.
<b><u>D</u>ECIMAL</b>	Places the data so that the decimal point in the data is at the tab position.

To tabulate data in a specific column, multiple tab characters are possible: issue the TABS line command and type over each asterisk (\*) marking the tab positions with another special character. Any input preceded by any of these special characters is tabulated in the corresponding column. You can type an L(EFT), R(IGHT) or D(ECIMAL) after each tabulation character to specify placement of data for the tab position.

## Examples of the TABS Command

The following examples assume the ampersand (&) to be the tabulation character:

**Example 1 - Tab Positions:**

The command

**TABS 10 20 40 LEFT**

activates logical tabs with tabulation columns 10, 20, and 40 with left justification. After you press ENTER, the input text line

```
&abc &def &ghi
```

is displayed as follows:

```
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6
          abc      def              ghi
```

**Example 2 - TABS RIGHT:**

The command

**TABS RIGHT**

activates logical tabs with right justification. After you press ENTER, the input text line

```
&abc &def &ghi
```

is displayed as follows:

```
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6
          abc      def              ghi
```

**Example 3 - TABS DECIMAL:**

The command

**TABS DECIMAL**

activates logical tabs with justification of the decimal point in the tab position. After you press ENTER, the input text line

```
&15.27$ &16.3 DM &13 IS
```

is displayed as follows:

```
=cols> ----+----1----+----2----+----3----+----4----+----5----+----6
          15.27$   16.3 DM              13 IS
```

**Example 4 - Mixed Justification:**

Issue the command **TABS 10 20 30 40 50**. Then issue the TABS line command. This displays the current tab positions as follows:

```
=tabs          *          *          *          *          *
```

Type an L, R or D next to each tab position as required (unmarked tab positions assume the value of the last TAB command):

```
=tabs          *R          *D          *D          *D          *L
```

After you press ENTER, the input text line

```
&start &0.01 &0.02 &0.03 &end
```

is displayed as follows:

```
=cols>  ----+----1----+----2----+----3----+----4----+----5----+----6
          start      0.01      0.02      0.03      end
```

**Example 5 - Multiple Tab Symbols:**

Type over the asterisks in the **=tabs** line with other special characters and specify left justification for each one as follows:

```
=tabs          ]L          &L          #L          $L          =L
```

After you press ENTER, the input text line

```
=first$second#third&fourth]fifth
```

is displayed as follows:

```
=cols>  ----+----1----+----2----+----3----+----4----+----5----+----6
                                                first
                                     second
                                third
                        fourth
                fifth
```

**Example 6 - Using a Blank as Tabulation Symbol:**

Issue the command

**TABS ' '**

which activates tabulation with one blank as tabulation character. This means that words separated by one blank are tabulated. After you press ENTER, the input text line

```

this is a blank tabulation
    
```

is displayed as follows:

```

=cols> ----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6
           this      is      a      blank      tabulation
    
```

**TOP**

```

TOP
    
```

This command is used to scroll to the beginning of the object being edited.

**UC**

```

UC [T] 'stringf'
   [C] 'stringf'
   [X] 'stringf'
   [P] 'stringf'
   [.X] [n]
   [.X.Y] [n m]
   [ALL]
   [NEXT]
   [PREV]
   [FIRST]
   [LAST]
   [CHARS]
   [WORD]
   [PREFIX]
   [SUFFIX]
   [NX]
   [X]
    
```

The UC command converts one or more lines to upper case. It applies the same parameters as the LC command. If you enter the UC command without parameters, it changes the current line to upper case.

**UNDO**

```

UNDO [ALL]
     [n]
    
```

If the log file is active (see the LOG command), the UNDO command backs out all changes made since the last time you pressed ENTER. Repeated use of the UNDO command backs out consecutive changes in reverse order. You can thus back out all changes one by one until you restore the member to its original status at session begin.

You can specify the following parameters with the UNDO command:

<b>ALL</b>	All modifications made in the current edit session are backed out.
<i>n</i>	The last <i>n</i> modifications are backed out.

**UP**

```
UP [n]
```

This command scrolls upwards in the source code.

The parameter *n* specifies the number of lines to be scrolled upwards. If *n* is omitted, the scroll amount is determined by the scroll mode.

**WINDOW**

```
WINDOW {
    line1 line2
    line1 line2 column1
    line1 line2 column1 column2
}
```

This command is used to define a data window to be copied or moved. The starting line and column and the end line and column of the window are specified in the command parameters. At least *line1* and *line2* are required.

<i>line1 line2</i>	Defines a window starting at column 1 of <i>line1</i> and ending in the last column of <i>line2</i> .
<i>line1 line2 column1</i>	Defines a window starting at <i>column1</i> of <i>line1</i> and ending at the last column of <i>line2</i> .
<i>line1 line2 column1 column2</i>	Defines a window starting at <i>column1</i> of <i>line1</i> and ending at <i>column2</i> of <i>line2</i> .

**X**

```
X
```

This command places the line marked with line command .X at the top of the data area.

**XSWAP**

```
XSWAP
```

This command is used to exchange displayed lines with excluded lines. Lines are excluded using the EXCLUDE command.

**Y**

```
Y
```

This command places the line marked with line command .Y at the top of the data area.

## Line Commands

You can enter a line command on any data line by typing over the line number on the left of your edit screen. A line command always applies to the line in which you enter it (or to a block of lines marked by multiple line commands).

When the insert mode is active, it is not possible to enter line commands in the line number field. Toggle the insert mode to non-insert mode.

Line commands can also be entered in command line at the top of the editor screen. In this case, the command must be preceded by a colon (:) and applies to the line marked by the cursor.

Command	Explanation
)	Moves this line right by two columns.
)n	Moves this line right by $n$ columns, irrespective of any other data in the line: you may lose data in the moved line.
))n	Marks first line of a block to be moved right by $n$ columns. A second ))n is required to mark the last line of the block. The block is moved regardless of any other data in the block: you may lose data in the moved block.
(	Moves this line left by two columns.
(n	Moves this line left by $n$ columns regardless of any other data (you may lose data in the moved lines).
((n	Marks first line of a block to be moved left by $n$ columns. A second ((n is required to mark the last line of the block.
<	Moves data in this line left by two columns.
>	Moves data in this line right by two columns.
>n	Moves data in this line right by $n$ columns (or up to last non-blank character: no data are lost).
>>n	Marks first line in a block to be moved to the right by $n$ columns (or until last non-blank character). A second >> is required to mark the last line of the block.
<n	Moves data in this line left by $n$ columns (or until first non-blank character).
<<n	Marks first line in a block to be moved to the left by $n$ columns (or until first non-blank character). A second << is required to mark the last line of the block.
A	Marks the target line for a move (M, Mn, MM) or copy (C, Cn, CC) line command. The moved/copied line(s) are inserted <i>after</i> this line.
B	Marks the target line for a move (M, Mn, MM) or copy (C, Cn, CC) line command. The moved/copied line(s) are inserted <i>before</i> this line.
BNDS	Displays the boundary positions in this line.
C	Copies this line to the position indicated by an A, B or O line command.
Cn	Copies the present line and the next $n-1$ lines to the position indicated by an A, B or O line command.
CC	Marks the first line of a block of lines to be copied. A second CC command is required to mark the last line of the block to be copied. The lines are copied to the position indicated by an A, B or O line command.

Command	Explanation
<b>CX</b>	Copies the line labelled .X. Inserts data after this line.
<b>CY</b>	Copies the line labelled .Y. Inserts data after this line.
<b>CX-Y</b>	Copies the block of lines from the line labelled .X to the line labelled .Y. Inserts data after this line.
<b>COLS</b>	Displays the column positions in this line.
<b>D</b>	Deletes this line.
<b>Dn</b>	Deletes the present line and the next <i>n-1</i> lines.
<b>DD</b>	Marks the first line of a block to be deleted. A second DD command is required to mark the last line of the block to be deleted. The deletion is performed after second the DD has been entered.
<b>DX</b>	Deletes the line labelled .X.
<b>DY</b>	Deletes the line labelled .Y.
<b>DX-Y</b>	Deletes the block of lines from the line labelled .X to the line labelled .Y.
<b>F</b>	Includes the first excluded line.
<b>Fn</b>	Includes the first <i>n</i> excluded lines.
<b>I</b>	<p>Inserts one line. The editor switches to insert mode. This means if you type data or enter a blank on the new line and press ENTER, a new line is automatically inserted and the cursor placed in it.</p> <p>If you enter no new data in an inserted line and press ENTER, the editor leaves insert mode and the blank line is deleted (see also the editor command EMPTY).</p> <p>You can also fill an inserted line with a predefined content (see the editor command MASK).</p>
<b>In</b>	Inserts <i>n</i> lines. You may type data in the new lines. When you press ENTER, unused lines are deleted but one blank line remains with the cursor in it (editor stays in insert mode).
<b>.I(obj,ssss,nnnn)</b>	<p>Inserts any object contained in the current library into the edit area. This command is entered in the editor area, not in the prefix area. The period preceding the "I" is the escape character. .I(*) invokes a selection list of objects in the current library.</p> <p>The "ssss" entry indicates the line at which the include operation is to begin. For example, setting "ssss" to 20 causes the insertion to begin with the 20th line in the program.</p> <p>The "nnnn" entry indicates the number of lines to be inserted.</p> <p>If the object is a Natural map, an INPUT USING MAP statement with all defined variables is automatically included in the current line.</p> <p><b>Note:</b> Only stowed data areas can be included into the source area.</p>
<b>J</b>	Joins next line with this one. You can specify how many of the characters of the following line are to be joined by placing the cursor at the point in the line where it is to be separated and press ENTER. To join the entire line, place the cursor outside the line to be joined. This command is identical to the TJ command.
<b>Ln</b>	Includes the last <i>n</i> excluded lines.
<b>LC</b>	Changes this line to lower case.
<b>LCn</b>	Changes the present line and the next <i>n-1</i> lines to lower case.
<b>LCC</b>	Marks the first line of a block to be changed to lower case. A second LCC is required to mark the last line in the block.
<b>LJ</b>	Justifies the data within the set boundaries in this line with the left boundary.

Command	Explanation
<b>LJJ</b>	Marks the first line of a block of data within the set boundaries to be justified to the left. A second LJJ command is required to mark the last line of the block to be justified. The justification is performed after the second LJJ command has been issued.
<b>M</b>	Moves this line to the position indicated by an A, B or O line command.
<b>Mn</b>	Moves the present line and the next $n-1$ lines to the position indicated by an A, B or O line command.
<b>MM</b>	Marks the first line of the block to be moved. A second MM command is required to mark the last line of the block to be moved. The lines are moved to the position indicated by an A, B or O line command.
<b>MASK</b>	Inserts a blank line in the editor into which you can create a mask. This line is inserted whenever the insert ( <i>In</i> ) line command is used to create one or more new lines (see also the editor command MASK).
<b>MX</b>	Moves the line labelled .X. Inserts it after this line.
<b>MY</b>	Moves the line labelled .Y. Inserts it after this line.
<b>MX-Y</b>	Moves the block of lines from the line labelled .X to the line labelled .Y. Inserts it after this line.
<b>N</b>	Modifications made in this line do not take effect when ENTER is pressed.
<b>NZ</b>	The line marked with this command is placed at the top of the editor area when a POINT editor command is issued.
<b>O</b>	Marks this line as target line for a move (M, Mn, MM) or copy (C, Cn, CC) line command. The moved/copied line(s) are merged with this line, that is, blank characters in the line are overlaid.
<b>On</b>	Marks the present line and the next $n-1$ lines as target lines for a move (M, Mn, MM) or copy (C, Cn, CC) line command. The moved/copied lines are merged with these lines, that is, blank characters in the lines are overlaid.
<b>OO</b>	Marks the first line of a block of target lines for a move (M, Mn, MM) or copy (C, Cn, CC) line command. A second OO command is required to mark the last line of the block of target lines. The moved/copied line(s) are merged with these lines, that is, blank characters in the lines are overlaid.
<b>R</b>	Repeats this line once.
<b>Rn</b>	Repeats this line $n$ times.
<b>RR</b>	Marks the first line of a block to be repeated. A second RR command is required to mark the last line of the block to be repeated. The repeat operation is performed after the second RR has been entered.
<b>RRn</b>	Repeats the block of lines $n$ times.
<b>RJ</b>	Justifies the data within the set boundaries in this line with the right boundary.
<b>RJJ</b>	Marks the first line of a block of data within the set boundaries to be justified to the right. A second RJJ command is required to mark the last line of the block to be justified. The justification is performed after the second RJJ has been issued.
<b>S</b>	Splits this line into two lines beginning at the cursor position. Type in the command, move the cursor to the position where the line is to be split, and press ENTER.
<b>T</b>	Scrolls the data to make the marked line the top line.

Command	Explanation
<b>TABS</b>	Displays the tab positions in this line.
<b>TC</b>	Centers the data within the set boundaries in this line.
<b>TCC</b>	Marks the first line of a block of data within the set boundaries to be centered. A second TCC command is required to mark the last line of the block of the centered. The centering is performed after the second TCC command has been issued.
<b>TE</b>	Switches editor to text enter mode (blank screen to end of screen).
<b>TF</b>	Joins this line with the following lines until the next blank line. The bounds settings can be used to restrict the columns affected (see BNDS command).
<b>TF<math>n</math></b>	This line command may be entered with a numerical value specifying the right boundary, e.g. the line command TF50 orders data with column 50.
<b>TI</b>	Inverts sequence of all characters in the current line and within the set boundaries.
<b>TII</b>	Marks the first line of a block of text to be inverted within set boundaries. Requires a second TII to mark the last line of the block.
<b>TJ</b>	Joins next line with this one. Same as the join (J) command.
<b>TO</b>	Joins this line with the next one.
<b>TOO</b>	Marks the first line of a block of data within the set boundaries to be joined. A second TOO command is required to mark the last line of the block to be joined. The function is performed after the second TOO has been issued.
<b>TS</b>	Splits this line into two lines at the cursor position; an empty line is also automatically inserted, but deleted if unused (identical to <b>S</b> line command).
<b>UC</b>	Changes this line to upper case.
<b>UC<math>n</math></b>	Changes the present line and the next $n-1$ lines to upper case.
<b>UCC</b>	Marks the first line of a block to be changed to upper case. A second UCC is required to mark the last line of the block.
<b>W</b>	Opens window with one line.
<b>W<math>n</math></b>	Opens window with $n$ lines.
<b>WC</b>	Copies the data window. The cursor position marks the column at which this line is to be split to insert the copied data.
<b>WC<math>n</math></b>	Splits this line in column $n$ , and copies the data between the two parts of the line.
<b>WE</b>	Marks end of data window. Works in the same way as WS. If the window is to start and end in the same line, overtype the the WS command with the WE command. The editor acknowledges the set window with message WW in the line command field.
<b>WM</b>	Moves the data window. Works in the same way as WC, but the original data are deleted after the copy operation.
<b>WM<math>n</math></b>	Splits this line in column $n$ , and moves the data between the two parts of the line.
<b>WS</b>	Marks start of data window. The cursor position marks the column from which data are read. If the cursor is not in the line for which the command is entered, column 1 is taken.
<b>WS<math>n</math></b>	Data window starts in column $n$ of this line.
<b>X</b>	Excludes this line.
<b>X<math>n</math></b>	Excludes the following $n$ lines.

Command	Explanation
<b>XX</b>	Marks the first line of the block to be excluded. A second XX is required to mark the second line of the block.
<i>.label</i>	Marks this line with " <i>.label</i> ". The <i>label</i> may be any string of 1 to 4 alphabetical characters. See also the editor command LABEL. For example: . X names this line . X. and .Y names this line .Y.

# Data Area Editor

- Invoking the Data Area Editor
- Editor Modes
- Editing Screen
- Command Mode
- Editor Commands
- Edit Mode
- Field Types
- Line Commands

The Natural data area editor is used to create and maintain local data areas, global data areas, and parameter data areas.

A data area may consist of user-defined variables, database views, and global data blocks (a collection of variables and/or views).

With the data area editor, you can create, change, save and stow data areas. The data areas can be checked for their syntactical correctness. You can also create copycode from a data area.

## Invoking the Data Area Editor

The editor is invoked by the EDIT command specifying the type of data area to be edited (possible abbreviations are underlined):

$\text{EDIT } \left\{ \begin{array}{l} \underline{\text{G}}\text{LOBAL} \\ \underline{\text{L}}\text{OCAL} \\ \underline{\text{P}}\text{ARAMETER} \end{array} \right\} [\textit{area-name}]$
--

If you specify no data area name, the data area editor will create a new data area of that type.

## Editor Modes

The data area editor operates in two different modes:

- Edit Mode  
In edit mode, the data area can be modified. Lines can be added, deleted or changed.
- Command Mode  
In command mode, data area editor commands such as `SAVE`, `STOW`, `CHECK`, etc. can be entered.

By default, the data area editor is in edit mode when you invoke it. To toggle from one mode to the other, you press `ESC`.

## Editing Screen

The data area editing screen (with a local data area in the edit area) is shown below:

```

                                Press <ESC> to enter command mode
Mem: PER-DISL  Lib: SYSEXP  Type: LOCAL  Bytes: 2352  Line: 0 of: 20
C T  Comment
*   *** Top of Data Area ***
V 1 PERS1                                PERSONNEL
  2 NAME                                A 20
  2 FIRST-NAME                          A 15
  2 INITIAL                              A 1
  2 SEX                                  A 1
  2 FAMILY-STATUS                        A 10
  2 NUMBER-OF-DEPENDENTS                 N 2.0
  2 NUMBER                                N 5.0
  2 STREET                                A 20
  2 CITY                                  A 15
  2 STATE                                A 2
  2 ZIP                                  N 5.0
  2 JOB                                  A 20
  2 SALARY                               N 6.0
  2 COMMISSION                           N 6.0
  2 YEARS-OF-EDUCATION                   N 2.0
  2 YEARS-WITH-COMPANY                   N 2.0
  2 VACATION-DAYS                        N 2.0
  2 SICK-DAYS                            N 2.0
F 1 HELP      F 2 CHOICE  F 3 QUIT    F 4 SAVE    F 5 STOW    F 6 CHECK
F 7 READ      F 8 CLEAR   F 9 MEM TYPE F10 GEN    F11 FLD TYPE F12
    
```

The editing screen of the data area editor is divided into five areas. These areas and their functions are listed below:

Area	Explanation
<b>Command Line</b>	The command line is used to issue editor commands. The command line consists of either a selection menu or a direct command line. Press ESC to move from editing to selection menu mode or the direct command line, whichever was most recently invoked. Use the "M" command to move from the direct command line to selection menu mode. Possible commands are described in the section Editor Commands.
<b>Status Line</b>	The status line contains the following information about the data area currently being edited: <b>Mem:</b> data area name <b>Lib:</b> current library <b>Type:</b> data area type (global, local, or parameter) <b>Bytes:</b> size of data area in bytes <b>Line:</b> number of current edit line <b>of:</b> total number of lines
<b>Edit Area</b>	In the edit area, the data area is displayed. The first line of the edit area is the edit header, which describes the current edit line. <b>Note:</b> In edit mode, the current edit line is highlighted.
<b>F-Key Lines</b>	Function-key assignments (edit mode only).
<b>Message Line</b>	This line displays error messages. An error message temporarily overwrites the first function-key line.

## Command Mode

In command mode, you can enter a Natural edit command, or the name of a Natural program.

When you enter command mode, the direct command line appears by default.

To get to the selection menu, you enter the option "M" (menu). A selection menu is displayed containing the following items:

- Commands
- Direct Command Line
- Quit

To leave the data area editor, you select "Quit".

**Note:**

The current data area will not be saved.

Once the selection menu has been invoked, it will always appear when command mode is entered. Function-key assignments are not used.

To return to command mode, select the direct command menu item.

**Note:**

Most data area editor commands can be entered in both menu and direct command mode.

## Editor Commands

When the editor is in command mode, the following edit commands can be entered in the command line of the data area editor or selected from the command menu:

Command	Function
<b>CHECK</b>	Checks the data area definition currently located in the edit source area for syntactical correctness. A window informs you that a syntax check is in process. If a syntax error is found, the line containing the error becomes the current line, an alarm sounds, and the error is displayed in the message line. If no errors are found, a corresponding message is displayed.
<b>CLEAR</b>	Deletes the current data area from the source area. However, it is not deleted from the library. Changes are lost if they were not previously saved.
<b>FLD TYPE</b>	This command is used to change the type of the actual data. A window appears containing the various data types: <b>D</b> - Data Field <b>B</b> - Block <b>C</b> - Constant <b>H</b> - Handle <b>S</b> - Structure <b>U</b> - Globally Unique ID <b>*</b> - Comment <b>Note:</b> The data type of fields within a view definition cannot be changed. Some field information can be lost if you change to another type. This command cannot be entered in direct command mode.
<b>GEN</b> <i>object-name</i>	Generates Natural copycode from the current data area. The copycode is generated into the source area, thus the data area is deleted from the source area. Changes made to the source area since the last SAVE or STOW are lost.
<b>READ</b> <i>object-name</i>	Reads a Natural object from a library into the edit source area. The object name must be specified.
<b>SAVE</b> <i>data-area</i>	Saves the data area currently in the source area. You are prompted for the data area and library. If the current name and library are correct, no entry is required. The definition is not checked before being saved.
<b>SET ABS [ON OFF]</b>	This command determines whether the SCAN command operates in absolute or non-absolute mode.  ON: the SCAN command operates in absolute mode, which means that the value to be scanned need not be delimited by blanks or special characters.  OFF: the SCAN command operates in non-absolute mode, which means that the value to be scanned must be delimited by blanks or special characters.  The default is OFF.

Command	Function
<b>SET SCAN</b> <b>COMMENT NAME LEVEL</b>	If SET SCAN is set to COMMENT, you can scan for a value in the "Comment" column. If SET SCAN is set to NAME, you can scan for a value in the "Name" column. If SET SCAN is set to LEVEL, you can scan within a hierarchical structure. You cannot scan in both columns simultaneously; the default is NAME.
<b>STOW</b> <i>data-area</i>	Before a data area can be used in a Natural program, it must be stowed. The STOW command saves and catalogs the data area currently in the edit source area. You are prompted for the data area and library. The data area is checked before being cataloged. If no errors are found, a corresponding message is displayed. LDAs and PDAs are saved as source code only.
<b>TYPE</b>	This command is used to change the data area type. A list appears containing the various types available: <b>G</b> Global <b>L</b> Local <b>A</b> Parameter

After the execution of any command, the data area editor switches automatically back to edit mode. Press ESC to return to command mode.

## Edit Mode

In edit mode, you can make changes to the data area. Changes can only be made on the current edit line which is highlighted.

The first line of the edit area may be accessed by using the HOME key. The last line of the data area may be accessed using the END key. The keys PAGE UP or PAGE DOWN can be used to move the edit line up or down one screen page at a time (Check your SAGtermcap file for entries if you have no access to any of these keys).

The edit area can also be scrolled using the UP ARROW and DOWN ARROW keys.

The displayed function keys can also be used with this mode.

The edit header will change according to the current edit line field type.

## Field Types

The data area editor recognizes the following field types:

Type	Explanation
<b>Data Field</b>	User-defined or database variable.
<b>Constant</b>	User-defined constant value. (C)
<b>Multiple Field</b>	Database field with more than one occurrence. (M)
<b>Structure</b>	Consists of data fields, constants and other structures (max. levels: 9). Also known as a group. (S)
<b>Database Group</b>	Database group. (G)
<b>Periodic Group</b>	Database group with more than one occurrence. (P)
<b>View</b>	Database view. (V)
<b>Block</b>	Block definition within GDA. (B)
<b>Redefine</b>	Redefinition of a Field. (R)
<b>Filler</b>	Filler character. (F)
<b>Counter Field</b>	Counter (C*) for a multiple-value field or periodic group.
<b>Comment</b>	Comment line. (*)
<b>Object Handle</b>	Handle to reference objects in Natural programs. (O)
<b>Globally Unique ID</b>	Identifier for interface and classes guaranteed to be unique across all possible networks. (U)

### Note:

Characters in parentheses are the abbreviations used to describe the field type.

The edit header changes according to field type. During editing, the first two columns displayed on the editor screen are the same for each edit line.

The first column (C) is used to enter the edit commands. The second column (T) is used to display the line field type.

Line information, based on field type, cannot always be displayed on one line.

Those types containing information exceeding one line are marked with an "X" in the last column (M). To display this information, the edit command SHOW can be used (see Line Commands).

## Line Commands

Line commands for the data area editor can only be entered in the first column of each line.

Pressing F2 produces a list of all available line commands.

The following line commands are available in the data area editor:

Command	Function
<b>C (Copy)</b>	Copies one or more lines into the clipboard, without deletion. If COPY is issued within an edit block, all lines within this block are copied to the clipboard area.
<b>D (Delete)</b>	Deletes one or more lines. If DELETE is issued within an edit block, all lines within this block are deleted. Deleted lines are placed in the clipboard. For additional information, see the line commands COPY and PASTE.
<b>E (Edit)</b>	Modifies/edits information for an existing field. Please note that changing the initialization type deletes all previously entered initialization information.
<b>H</b>	Deselects the current edit block and must be issued from within the edit block.
<b>I (Insert)</b>	<p>Inserts a new field after the current line. A list displaying possible field types appears. After the desired type has been chosen, a window appears in which you can enter all information necessary for the selected field type. If field initialization is possible and desired, the initialization type must be entered. There are two initialization types:</p> <p><b>Free form</b> - (enter F in the initialization field) initialization requires you to enter a complete initialization statement as defined by Natural (DEFINE DATA statement).</p> <p><b>Single value</b> - (enter S in the initialization field) initialization requires you to enter the value of the field only. If the field is an array, all elements of the array are listed. A value for each element can be entered (optional). Values are entered based on field type.</p> <p><b>Note:</b> Parentheses, apostrophes or value prefixes (for example, H=Hex, D=Date or T=Time) are not required.</p>
<b>P (Paste)</b>	Inserts lines from the clipboard into the data area after the current line. The contents of the clipboard are not deleted, thus they can be pasted more than once. Clipboard contents are changed when a COPY or DELETE command is issued or when a new data area is edited.
<b>R (Redefine)</b>	Redefines an object. The data area editor automatically creates a redefine line and a redefine window appears. The data area editor keeps track of the number of free bytes still available for redefinition. If there are no free bytes, the redefine function ends.
<b>S (Show)</b>	Displays a window which contains all information about the selected field. Changes are not possible. If an initialization has been specified, a separate window containing initialization information is also displayed. Information scrolling is possible.

Command	Function
<b>V (Define view)</b>	<p>Invokes a window to define a view. The window contains the following items:</p> <ul style="list-style-type: none"> <li>- Name of view</li> <li>- Name of DDM</li> <li>- Comment</li> </ul> <p>After these names have been entered, the DDM is displayed. You can scroll through the DDM with the cursor and select the fields which are to be included in the data area by marking them with:</p> <ul style="list-style-type: none"> <li><b>X</b> - to select individual fields</li> <li><b>A</b> - to select all fields</li> </ul> <p><i>blank</i> to select individual fields that are not to be included if you have also specified "A".</p> <p>If you select a periodic group or multiple-value field, you are prompted to supply the number of occurrences.</p>
<b>X</b>	Marks the beginning of an edit block.
<b>Y</b>	Marks the end of an edit block.
<b>Z</b>	<p>Marks an entire structure as an edit block.</p> <p>The edit block starts at the current line and all continuous lines with levels less than the current line are marked. For example, if "Z" is entered in a view line the entire view is marked as the edit block.</p>
<b>*</b>	Creates a C* variable (internal count of occurrences) for a multiple-value field or periodic group.

The steps used to move a block to the clipboard and return it to the Data Area Editor (Block Move), are as follows: Mark the first line as "X" and the last line as "Y" and use the "C" command from within the marked block to move it to the clipboard. Position the cursor and use the "P" line command to paste the lines from the clipboard to below the cursor position.

# Map Editor

- Summary of the Map Creation Process
- Map Fields
- Invoking the Map Editor
  - Creating a New Map
  - Editing an Existing Map
- Map Editor Menu
- Creating a Text Constant
- Creating a User-Defined Variable
  - Using Natural System Variables in a Map
- Modifying a User-Defined Variable - Field Editing
  - Rule Editing - Processing Rules
  - Array Editing
  - AD - Attribute Definition
- Selecting Fields from a DDM or User View
- Defining Fields for a Parameter or Local Data Definition
  - Parameter Definitions
  - Local Data Definitions
- Map Profile
  - Map Profile Settings
  - Filler Characters
- Post Assignment
- Field-Sensitive Processing
  - Advantages of Field-Sensitive Processing
  - Defining a Map as Field-Sensitive

The Natural map editor is used to create maps (screen layouts).

A map can be stored in a Natural library, from where it can be invoked by a Natural program using an INPUT USING MAP statement (for input maps) or a WRITE USING MAP statement (for output maps).

## Summary of the Map Creation Process

There are four major steps involved in the creation of a map:

1. Definition of the map profile (that is, the format settings and filler characters to be used). A menu is provided from which you select the desired items.
2. Definition of the map. A map can be created in two different ways:
  - First create a prototype map, next create the corresponding data views, then integrate the map into the application.  
Fields can be defined directly on the screen. Each field is assigned a default name. Subsequently, when the corresponding data views have been created, the actual field definitions can be assigned to the map fields.
  - Create a map using existing data views.  
If data views already exist, the map fields can be created by using the field definitions contained in the data views. In this case, all characteristics of a field defined in the data views are included when the field is positioned on the screen.
3. Definition of the fields to be used in the map. A full set of map editing facilities is provided which permit simple and efficient field definition.
4. The saving and/or cataloging of the map definition. Once defined, the map can be saved in source form and/or object form in a Natural library. Once saved, a map definition can be read and modified during a subsequent map editor session. Once stowed, a map definition can be invoked from a Natural program.

## Map Fields

A map consists of fields. A field can be either a text constant or a data variable. The fields which are to comprise a map definition can be specified in any of the following ways:

- The field can be defined directly on the screen as a text constant, or a user-defined variable.
- The field can be selected from a user view or data definition for any existing Natural object type in the library.
- A Natural system variable can be used.

Data variables can be either system variables or data copied from a view or data area. These variables can also be user-defined or copied from other source objects.

The following sections describe each of these methods in detail.

## Invoking the Map Editor

You invoke the Natural map editor to create a new map or edit an existing one as follows:

### Creating a New Map

On the Natural Main Menu, select "Direct" and press ENTER to invoke the Direct Command window.

**Note:**

You can also invoke the Direct Command window by selecting the <DIRECT COMMAND> from the library selection list that is displayed when you select "Library" from the Natural Main Menu.

In the Direct Command window, enter the following command:

**EDIT MAP or in short form: E M**

Press ENTER. The Map Editor Menu will be displayed:

```

.....
                                NATURAL MAP EDITOR (Esc to select field)      .
Create   Modify   Erase   Drag   Info OFF  Lines   Ops. Map  Quit      .
.....

```

### Editing an Existing Map

On the Natural Main Menu, select "Direct" and press ENTER to invoke the Direct Command window.

In this window, enter the following command:

**EDIT *map-name* or in short form: E *map-name***

If you do not remember the name of the map you wish to edit, select "Library" on the Natural Main Menu and press ENTER. A list of all available libraries will be displayed.

From the list, select the desired library with the cursor, and press ENTER. A list of all objects in this library will be displayed.

**Note:**

File designations are listed under the column "Pgm. Type".

With the cursor keys, scroll through the list until the desired map appears on the list. Then mark it with an "E" (Edit) in the column before the object names and press ENTER. The map editor will then be invoked for the selected map.

**Note:**

The F2 key invokes a window listing valid functions (for example, C=Check, D=Read, E=Edit) for this specific object for the item Ops Map.

Regardless of the map invoked, you are prompted at the bottom of the screen to use the cursor keys to select a field and to press ENTER. The selected field is highlighted.

**Note:**

The rightmost position of the bottom line displays the column and row number of the highlighted field.

Press ESC and the Map Editor Menu is invoked.

**Note:**

To change the programming mode (structured/reporting) for an existing map, enter the following commands in the Direct Command window:

READ map-name

GLOBALS SM=ON/OFF

SAVE map-name.

## Map Editor Menu

The Map Editor Menu, as shown above, is the main menu of the map editor. The following items can be selected from this menu:

- Create
- Modify
- Erase
- Drag
- Info ON/OFF
- Lines
- Ops. Map
- Quit

### Create

If you press ENTER, a list containing the following items is displayed:

- **A** - Parameter Data Area
- **G** - Global Data Area
- **H** - Helproutine
- **L** - Local Data Area
- **M** - Map
- **N** - Subprogram
- **P** - Program
- **S** - Subroutine
- **T** - Text Constant
- **U** - User Defined
- **V** - View Defined
- **1** - Parameter Defined
- **2** - Local Defined

These items are used to select fields/variables from an object for an object of this class. The object class can also be invoked by entering "C" and the object class abbreviation (key-sensitive). For more detailed information, see the section Selecting Fields from a DDM or User View.

### Modify

Modify enables you to modify a selected field. The selected field is the current field and it is highlighted.

A window displaying field attributes (extended field editing) is displayed in which the contents of these attributes can be modified.

### Erase

Erase enables you to delete the current field. You are then prompted "Delete field (Y/N)?"

All responses are key-sensitive. Caution is recommended!

If the current field is an array field, the entire array (not just the field) will be deleted.

## Drag

Drag enables you to move the current field to any unoccupied position on the screen. The selected field can be moved without restriction, using the cursor.

Once the field is positioned and ENTER is pressed, this field position takes effect.

## Info ON/OFF

Info ON/OFF is used to switch the display of the field information window ON and OFF (toggle switch). OFF is the default value.

The ENTER key or "I" (key-sensitive) is used to switch between ON and OFF.

## Lines

Lines invokes a selection list from which you can select the following line-specific functions:

```
Insert After
Erase Line
Copy After
Duplicate Line
Move After
Split Line
Join Line
```

These functions can be selected to perform operations on an entire line (not a single field) in a map. All operations are performed on the current line.

### Note:

These functions are self-explanatory and prompts appear at the bottom of the screen for each item selected.

## Ops. Map

Ops. Map (map operations) invokes the following selection list:

```
C Check Map
E Edit Map
K Key Rules
L List Map
P Prof. Map
R Read Map
S Save Map
T Test Map
W Stow Map
```

The list contains the following items:

<b>Check Map</b>	Causes syntax checking and generation of source code.
<b>Edit Map</b>	Invokes the map editing screen for modification of an existing map definition. The map editor will start a new edit session.
<b>Key Rules</b>	Invokes editing of function-key-related processing rules.
<b>List Map</b>	Generates source code and lists it.
<b>Prof. Map</b>	Invokes a map profile window, which is described under Map Profile.
<b>Read Map</b>	Invokes the map editing screen to read an existing map definition.
<b>Save Map</b>	Performs a source code generation check and then saves the map. The map definition is saved in source form in the Natural library.
<b>Test Map</b>	The current map definition is tested to ensure that it can be executed successfully. This includes testing of all processing rules and help facilities.
<b>Stow Map</b>	Performs a source code generation check, as well as a save and catalog of a map definition. The map definition is cataloged and also saved in source form in the current Natural library.

## Quit

Quit terminates the map editor session.

If you have edited the map (not saved), selected "Quit" and pressed ENTER, the following prompt appears:

```
Modifications have not been saved, quit anyway Y/N ?
```

### Note:

Replies are key-sensitive; caution is recommended.

If the editor session is terminated, the Natural Main Menu is redisplayed.

## Creating a Text Constant

Select "Text Constant" from the Create list and press ENTER.

Depending on the map type invoked, a screen appears.

At the bottom of the screen, you are always prompted to position the cursor and enter text.

Position the cursor to the start of an empty field.

**Note:**

You cannot overwrite existing fields.

Enter the text. The first character entered causes the line to be highlighted. Highlighting indicates the maximum space available for text entry. Characters can be entered or deleted until you press ENTER.

**Note:**

The ESC key cancels text entry.

Press PF2 to select an attribute and color to be used for the text entered. Use the UP/DOWN ARROW keys to scroll through and select one of the available attributes/colors or simply use the corresponding abbreviation (for example, B=Blinking or RE=Red) and press ENTER. Using the LEFT/RIGHT ARROW keys, you can toggle between attribute and color definition.

Text entry is now complete.

**Note:**

If you want to create and/or define a data variable after having defined the text, see the section Using Natural System Variables in a Map.

## Creating a User-Defined Variable

Select "User Defined" from the field list and press ENTER.

The following window appears:

```

Extended Field Editing
Field :
Format: A Len:          AL:          PM:          ZP: N   SG: N
Rules : 0 Rule Editing: N Array:      Array Editing: N Mode:
AD:          CD:          CV:          DY: N   HE: N
EM:

```

A message appears at the bottom of the screen, prompting you to:

```

Position cursor and press Enter or format char.

```

Position the cursor to the start of a field position and press ENTER.

A line containing the maximum available length is displayed as well as a selection list of the valid data variable types.

### Note:

If the data type is known, the type character can be entered directly. The display of available data types is thus avoided.

Select the data variable and press ENTER. The field attributes window appears in the bottom portion of the screen.

The format is now the data type entered and a default name (for example, "#1") is assigned to the field.

There are two length fields displayed on this screen:

1. ... AL (Alphanumeric length) The display length.
2. Len The internal length of the data type.

The lengths of user-defined variables are defined by performing the following steps:

1. Enter the length of the first field (for example, Alphan. Len... ).
2. When the field length definition is complete, press ENTER.
3. You are then prompted to enter a field name for the variable. Having selected the appropriate field name, press ENTER.

The definition function for the first field is now complete.

The cursor moves automatically to the second length field (Len... ). Change the length field definition or use the TAB key. Continue this process to define all other pertinent information to be used for the field being defined.

For further information on these fields, see [Modifying a User-Defined Variable - Field Editing](#).

When this definition is complete, press ENTER.

## Using Natural System Variables in a Map

Natural system variables can also be specified in a map definition.

A Natural system variable can be selected with Create user-defined field.

Select "\*" from the field list and select the system variable from the list provided.

The format of the specified system variable is inserted in the field definition form.

## Modifying a User-Defined Variable - Field Editing

The map editor is used to define a field with all its attributes.

Select the item Create or Modify on the field editing selection menu and press ENTER. The following window displaying field attributes for the current field appears:

Extended Field Editing					
Field :					
Format:	A	Len:	AL:	PM:	ZP: N SG: N
Rules :	0	Rule	Editing: N	Array:	Array Editing: N Mode:
AD:		CD:	CV:		DY: N HE: N
EM:					

With the field attributes window, any selected field can be modified. A selected field is the current field, which is highlighted.

Entry	Explanation
<b>Field</b>	<p>The field name. Field name assignment is related to the method with which the field was originally defined.</p> <p>If the field was taken from a user view or data definition, it is assigned the same name as the field in the user view or data definition.</p> <p>If the field was specified as a Natural system variable, it is assigned the name of the specified variable.</p> <p>If the field is neither of the above, it is assigned a dummy name. You must assign a name to such a field prior to map execution.</p> <p>The name of a field can be changed. However, a prefix must not be used for a field which did not have a prefix assigned previously. To obtain a prefixed field name, select the field from a user view or data definition. You are prompted to enter a name. If modifications have been made you must press ENTER to continue. Otherwise, you can move through the field attributes using the TAB key.</p> <p><b>Note:</b> Duplicate field names are only allowed for fields defined as "output only fields".</p>
<b>Format</b>	The format and length of the field. These can be changed by overwriting the current entry.
<b>Len</b>	The internal program length of the variable.
<b>AL or NL or FL or DF</b>	<p>The length to be used when displaying the field. What is displayed in this field depends on the entered format.</p> <p>AL - (alphanumeric length) for formats alphanumeric, logical, time and system;  NL - (numeric length) for formats binary, integer, numeric, packed numeric;  FL - for format floating point;  DF - for format date.</p>
<b>PM</b>	<p>Print Mode:</p> <p><b>C</b> - indicates that an alternative character set is to be used (special character table as defined by you or the designated Natural administrator).</p> <p><b>I</b> - indicates inverse print direction.</p> <p><b>C I</b> - indicates normal print direction</p> <p>For Numeric fields only:</p> <p><b>ZP</b> - N indicates that zero values for the field are not to be printed. Y - indicates that zero values are to be printed.</p> <p><b>SG</b> - N indicates that no sign position is to be allocated. Y - indicates that a sign position is to be allocated.</p>
<b>Rules</b>	The number of processing rules currently defined for the field.

Entry	Explanation
<b>Rule Editing</b>	Editing of processing rules; you are prompted: "Inline Processing Rule Editing (Y, <PF2>=EDIT)?". <b>Note:</b> The source code used to define a processing rule is entered/edited in the same way as with the Natural program editor.
<b>Array</b>	Indicates whether the field is an array or not ("blank").
<b>Array Editing</b>	Editing of arrays; you are prompted: "Array Editing (Y, <PF2>=EDIT)?".
<b>Mode</b>	Indicates how the field was created: <b>DATA</b> - Field was created by selecting a field from a DEFINE DATA definition. <b>SYS</b> - Field is a system variable. <b>UNDEF</b> - Field was created directly on the screen and has a dummy name. <b>USER</b> - The name of the field was created by extended field editing. <b>VIEW</b> - Field was selected from a view.
<b>AD</b>	Field attributes; you are prompted: "Attribute definitions (<PF2>=EDIT)". An Attribute definition window, containing the following items is displayed: - Representation - Alignment - I/O Characteristics - Mandatory Characters - Length Characteristics - Upper/Lower Case - Filler Character This function incorporates a toggle feature. If the attribute definition character displayed is correct, use the ESC key and the character is not changed. To change the attribute characteristics of a field, select the desired attribute and press ENTER; the modification is inserted in the AD definition. Press ESC to exit this function. Each item selected invokes an attribute window. These windows are self-explanatory.
<b>CD</b>	Colour attributes; you are prompted: "Color definitions (<PF2>=EDIT <CSR-UP/DN>=Select (Esc=Cancel Enter=OK)); i.e. press PF2 to edit the color definition, use the cursor to select a color and confirm with ENTER.
<b>CV</b>	Control variable for dynamic field attributes; you are prompted: "(<PF2>=Edit Rank if Array) ". The name of a variable which contains the attributes to be used for this field. This variable must be defined with format C in the program. The control variable also contains a MODIFIED data tag, which indicates whether the field has been modified following map execution. A single control variable can be applied to several map fields, in which case the MODIFIED data tag is set if any of the fields referencing the control variable has been modified.
<b>DY</b>	Dynamic string attributes; you are prompted: "(Y,<PF2>=Edit)". This parameter is used to define certain characters contained in the text string of an alphanumeric variable to control the attribute setting. See the session parameter DY.

Entry	Explanation
<b>HE</b>	<p>The name of a help routine to be assigned to the field; you are prompted: "(Y, &lt;PF2&gt;=Edit)". A window is displayed which provides sufficient space to specify multiple parameters.</p> <p>This option consists of two operands:</p> <p>One which specifies the name of the help routine to be invoked. It can be specified as a text constant or as a user-defined variable which contains the name of the help routine;</p> <p>One which consists of up to 20 parameters which can be passed to the help routine. The parameters can be specified as constants or as user-defined variables which contain the values of the parameters. If an "=" is specified as a parameter, the name of the field as defined in the map definition is passed to the help routine. In the case of a help routine which is assigned to a map, "=" denotes the name of the map.</p> <p>As no explicit DEFINE DATA PARAMETER statement can be specified in the map editor, the format/length of the second operand is defined in the following way:</p> <ul style="list-style-type: none"> <li>- If a parameter specified as second operand in the HE option is defined as a field of a map, the parameter will reference this field.</li> <li>- If no field with that name exists, the parameter field must be defined as N7 (default format assumed) in the program that uses the map.</li> </ul> <p><b>Note:</b> For a detailed explanation of the operands used in the HE option, see the session parameter HE.</p>
<b>EM</b>	Edit mask to be used for the field.

Three of the above items are of special interest:

- Rule Editing
- Array Editing
- AD (Attribute Definition)

## Rule Editing - Processing Rules

### Field-Related Processing Rules

Three types of processing rules can be defined:

- Inline processing rules
- Free Predict rules
- Automatic Predict rules

Inline processing rules are defined within a map source and do not have a name assigned. The availability of Predict is not required for inline rules.

**Note:**

Field-related inline processing rules can also be executed on a field-by-field basis; for further information, see the section Field-Sensitive Processing.

Free Predict rules have a name assigned and are stored in the Predict Dictionary. You cannot modify an existing free Predict rule (this can only be done in Predict); however, you can read a free rule into the editor, modify it, and store it under a different name to create a new free rule.

Free Predict rules can also be defined by accessing Predict on a remote OpenVMS, UNIX or mainframe server. To be able to do so, you must have set up your NATPARM parameter file accordingly and established a corresponding link by using Natural RPC. For information on how to assign dictionary servers and how to use Natural RPC, refer to your Installing and Setting Up Natural on UNIX or Installing and Setting Up Natural on OpenVMS.

Inline rules can become free Predict rules (and vice versa) if you assign/remove the rule name.

Predict automatic rules apply to database fields and are defined by the Predict administrator. If a field is created by selecting it from a view or a data definition, and if the field is a database field, all automatic rules for that field are linked to the map definition. All automatic rules are concatenated and treated as a single map rule.

The rank of the automatic rules is defined in the map profile settings (default 1).

Automatic rules cannot be modified using the map editor. They can, however, be assigned a different rank by either using the command "P=n" or just overwriting the old rank.

If Predict rules are modified subsequently by the Predict administrator, or new automatic rules are linked to a database field, or automatic rules are removed, it is sufficient to recatalog the map.

An ampersand "&" within the source code of a processing rule is dynamically substituted with the fully qualified name of the field using the rule.

#### Example:

```
IF & = ' ' REINPUT 'ENTER NAME' MARK *&
```

### Function-Key-Related Processing Rules

Two types of function-key-related processing rules can be defined:

- Inline processing rules
- Free Predict rules

Function-key-related processing rules can be used to assign activities to program sensitive function keys during map processing. For function keys which already have a command assigned by the program, this command is executed without any rule processing.

#### Example:

```
IF *PF-KEY = 'PF3'
    ESCAPE ROUTINE
END-IF
```

When this rule is executed, map processing is terminated without further rule processing.

### Processing-Rule Ranks

A field can have up to 100 processing rules (Rank 0 to 99). At map execution time, the processing rules are executed in ascending order by rank and screen position of the field. PF-key processing rules are always assumed to have the first screen position.

For optimum performance, the following assignments are recommended when assigning ranks to processing rules:

Rank	Processing Rule
0	Termination rule
1 - 4	Automatic rules
5 - 24	Format checking
25 - 44	Value checking for individual fields
45 - 64	Value cross-checking between fields
65 - 84	Database access
85 - 99	Special purpose

### Processing-Rule Editing

To edit field-related processing rules, select "Rule Editing" on the Extended Field Editing window. To edit function-key-related processing rules, select "Key Rules" in the "Ops. Map" window.

The following window containing the options Rules and Fields appears:

```

..Current Field: PERSONNEL.STREET.....
.
.                R U L E   E D I T I N G (Esc = Quit)                .
. Rules                                Fields                                .
.....

                                Hobby:  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Address
-
State      :  XX
Zip        :  99999
City       :  XXXXXXXXXXXXXXXXXXXX
Street/Number: XXXXXXXXXXXXXXXXXXXX  99999

Create or modify a rule for this field
    
```

Select one of these options with the cursor and press ENTER.

If you select Fields, a list of variables used in the current map appears (for information purposes only).

If you select Rules, a selection list of existing rules for the current field appears.

On each list, the Predict rules are identified by their names, the inline rules by their first three source code lines.

There are the following ways to define processing rules for a field or key:

- Create a new processing rule
  - define a new rule
  - modify an existing rule and save it under a new name
- Assign an existing rule and

- edit or
- move or
- copy or
- unlink the rule.

**To define a new processing rule**

1. Select Create.

An empty rule editor is displayed.

2. Enter the rule. (Use source code in the same way as in the Natural program editor.)

3. If you want the rule to be a free Predict rule, name it. If you want the rule to be an inline rule, do not name it before saving it.

4. Save the rule, see the section Commands for Processing Rule Editing.

Important: Once you have saved the rule, you can only modify it in Predict.

**To modify an existing processing rule**

1. In the Rule Editor header enter the name of a Predict rule in the field Rule and press ENTER. The rule is displayed in the rule editor.

2. Modify the rule. (Use source code in the same way as in the Natural program editor.)

3. Rename the rule and save it, see section Commands for Processing Rule Editing.

Important: Once you have saved the rule, you can only modify it in Predict.

**To assign an existing rule**

1. Select an existing rule from the list.

2. Press ENTER.

A window with the following options is displayed: edit, move, copy, unlink.

- Edit  
Select edit to modify the rule. The name and contents of the rule (if it is a free rule) are displayed in the editor. See Step 2 of the section to modify an existing processing rule.
- Move  
Select move to modify the rule's rank. When you press ENTER a list is displayed from which you can select the new rank.
- Copy  
Select copy to copy the rule but assign it a new rank. When you press ENTER a list is displayed from which you can select the new rank.
- Unlink  
Select unlink to remove the rule from the field.
- Quit with ESC.

**Note:**

If rules are written referencing a database statement, a label should be used, not a line reference number.

After the desired field processing rule has been entered, issue the command "P=*nn*" (where *nn* is the processing rule rank). This command saves the rule automatically.

**Commands for Processing Rule Editing**

In the processing rule editor, processing rules can be selected for editing by using the following commands in the editor command line:

Cmd.	Function
<b>P<math>mn</math></b>	Select rule with rank $mn$ .
<b>P*</b>	Select rule from selection list.
<b>P</b>	Advance to next rule defined for the field.
<b>P=<math>mn</math></b>	Assign rule on current rank to rank $mn$ and save automatically.
<b>U</b>	Unlink (Delete)
<b>.</b>	End processing rule editing and save the rule.

**Array Editing**

To invoke array editing, use the TAB key to select the item "Array Editing?" on the field attributes window. Replace the character "N" with "Y" (key sensitive) or press PF2 and the Array Definition window appears:

<b>•Array Definition•</b> .....			
Name #1		Upper Bnds	1____ 1____ 1____
-----			
Dimensions	Occurrences	Starting from	Spacing
1 . Index vertical	1__	_____	0 Lines
0 . Index horizontal	1__	_____	1 Columns
0 . Index (H/V) V	1__	_____	0 Cls/Ls
.....			

This window displays the fields as defined on the map. If changes are not required, press TAB to proceed to the next field; if you want to return to a previous field, press SHIFT+TAB.

The Array Definition window contains the following entries:

Entry	Explanation
<b>Upper Bnds</b>	Indicates the upper bounds of the array; that is, the highest occurrence in (from left to right) the first, second and third dimension. If a field defined in a program is used to define the map array, the upper bounds of that field (user-defined variable or database field), as defined in the program, are used; these cannot be overwritten on the array definition screen. If the map array is derived from a user view array or a data definition, the dimensions of the map array must not exceed the dimensions shown in this field. If the map array is not derived from a user view array or a data definition, the dimensions of the map array must not exceed the dimensions as defined in the Natural program.
<b>Dimensions</b>	An array can have up to three dimensions. The order in which the dimensions of the array are mapped to the map layout is determined by the values entered to the left of the Index operands; the abbreviations used are: H=Horizontal and V=Vertical.
<b>Occurrences</b>	The number of occurrences to be defined for a dimension.
<b>Starting From</b>	The starting index value for a dimension. A numeric value can be used, or a variable name can be used to indicate that the actual value is supplied in the Natural program which invokes the map definition.
<b>Spacing</b>	The number of blank lines (for vertical dimensions) or blank columns (for horizontal dimensions) to be inserted between each dimension occurrence.

Enter the desired information and press ENTER. You are returned to the next item ("AD") listed on the field attributes window.

### Examples of Array Definitions

#### Example 1:

A one-dimensional array consisting of 10 vertical occurrences with 2 blank lines to be inserted between each occurrence.

```

•Array Definition.....
Name #1                               Upper Bnds 10__ 1__ 1__
-----
Dimensions      Occurrences  Starting from  Spacing
1 . Index vertical      10_           _____  2  Lines
0 . Index horizontal    1__           _____  1  Columns
0 . Index (H/V) V      1__           _____  0  Cls/Ls
.....
    
```

#### Example 2:

Same as example 1 except that the array is to be horizontal.

```

•Array Definition.....
Name #1                               Upper Bnds 10__ 1__ 1__
-----
Dimensions      Occurrences  Starting from  Spacing
0 . Index vertical      1__           _____  0  Lines
1 . Index horizontal    10_           _____  1  Columns
0 . Index (H/V) V      1__           _____  0  Cls/Ls
.....
    
```

**Example 3:**

A two-dimensional array. The first dimension consists of 10 vertical occurrences with 1 blank line between each occurrence. The second dimension consists of 5 horizontal occurrences with 2 blank columns between each occurrence.

•Array Definition•			
Name #1		Upper Bnds	10__ 5__ 1__
-----			
Dimensions	Occurrences	Starting from	Spacing
1 . Index vertical	10_	_____	1 Lines
2 . Index horizontal	5_	_____	2 Columns
0 . Index (H/V) V	1_	_____	0 Cls/Ls
.....			

**Example 4:**

Same as example 3 except that the order of the dimensions is reversed.

•Array Definition•			
Name #1		Upper Bnds	5__ 10__ 1__
-----			
Dimensions	Occurrences	Starting from	Spacing
2 . Index vertical	10_	_____	1 Lines
1 . Index horizontal	5_	_____	2 Columns
0 . Index (H/V) V	1_	_____	0 Cls/Ls
.....			

**Example 5:**

A three-dimensional array. The first dimension consists of 3 vertical occurrences with 1 blank line between each occurrence. The second dimension consists of 5 horizontal occurrences with 2 blank columns between each occurrence. The third dimension consists of 2 occurrences, expanded vertically within each occurrence of the first dimension.

•Array Definition•			
Name #1		Upper Bnds	3__ 5__ 2__
-----			
Dimensions	Occurrences	Starting from	Spacing
1 . Index vertical	3_	_____	1 Lines
2 . Index horizontal	5_	_____	2 Columns
3 . Index (H/V) V	2_	_____	0 Cls/Ls
.....			

**Example 6:**

An example using "Starting from". The first dimension consists of 10 vertical occurrences starting from index I. 'I' is defined in the map editor with format/length N7 by default. The second dimension consists of 5 horizontal occurrences starting from the index 3.

```

•Array Definition.....
Name #1                               Upper Bnds 10___ 5___ 1___
-----
Dimensions                             Occurrences   Starting from   Spacing
1 . Index vertical                      10_           1_____       1   Lines
2 . Index horizontal                    5_           3_____       2   Columns
0 . Index (H/V) V                       1_           _____       0   Cls/Ls
.....
    
```

**Example 7:**

An example of making a two-dimensional display from a one-dimensional array. The array consists of 40 elements. It is displayed in two columns with 20 lines each. This is achieved by specifying 0 as the horizontal index.

```

•Array Definition.....
Name #1                               Upper Bnds 40___ 1___ 1___
-----
Dimensions                             Occurrences   Starting from   Spacing
1 . Index vertical                      20_           _____       0   Lines
0 . Index horizontal                    2_           _____      10   Columns
0 . Index (H/V) V                       1_           _____       0   Cls/Ls
.....
    
```

**AD - Attribute Definition**

Attribute definition editing is performed as follows. Use the TAB key to move to the AD item in the field attribute window.

Press PF2 to invoke the following Attribute Definition window:

```

Children      : 99                      Years-Educ: 99
Family Status: XXXXXXXXXXXX           Years-Comp: 99
Sex           : X                      Vacation-D: 99
                                           Sick-Days : 99

                                           Hobby:  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Address
-
State        : XX                      •Attribute Definition•
Zip          : 99999                   •Representation   •
City         : XXXXXXXXXXXX           •Alignment        •
Street/Number: XXXXXXXXXXXX           •I/O Characteristics •
                                           •Mandatory Characters •
                                           •Length Characteristics •
                                           •Upper/Lower Case •
                                           •Filler Character  •

•Extended Field Editing.....
•Field= PERSONNEL.STREET
•Format= A Len= 20   Alphan. Len= 20   PM=
•Rules:  0 Rule Editing? N Array:      Array Editing? N      Mode= Data
•AD= FHWOIL      CD=      CV=                      DY= -> HE= ->
•EM=
.....
    
```

Select the desired item with the cursor keys.

Press ENTER and an additional selection window for each item selected appears.

**Note:**

This function has a toggle feature. If the attribute definition character displayed is correct, use the ESC key and this character is not changed. The ESC key terminates AD editing.

## Selecting Fields from a DDM or User View

New fields can be created by selecting existing variable definitions from a DDM or a user view. A field can be selected from any available DDM.

To select a DDM/user view, first select the object class. Valid object classes are:

<b>A</b>	Parameter Data Area
<b>G</b>	Global Data Area
<b>H</b>	Helproutine
<b>L</b>	Local Data Area
<b>M</b>	Map
<b>N</b>	Subprogram
<b>P</b>	Program
<b>S</b>	Subroutine
<b>V</b>	View

Programs, subroutines, subprograms and helproutines can only be used if they contain a DEFINE DATA statement.

For demonstration purposes, the object "view" will be used. The leftmost item on the Field Editing selection menu displays a list. Either select the item "View Defined" from this list and press ENTER or enter "C" and then the abbreviation "V".

The following library list appears on the screen:

```

.....
.                NATURAL MAP EDITOR (Esc to select field)                .
.Create  Modify  Erase   Drag   Info OFF  Lines   Ops. Map  Quit  .
.....
N. A Parameter Data Area .XXXX. ACTIO          . : XXXXXXXXXXXXXXXXXXXXXXXX
I. G Global Data Area   .      .ACTION          . : 999999
F. H Help Routine      .XXXX. AEH-BEDIENSTETER .ion: 999999
C. L Local Data Area   .      .AEH-HDAT          .duc: 99
F. M Map                .      .BED              .omp: 99
S. N Subprogram        .      .EMPLOYEES         .n-D: 99
. P Program            .      .EMPLOYEES-FILE    .ys : 99
. S Subroutine         .      .FUNC              .
. T Text Constant     .      .FUNCTION         .XXXXXXXXXXXXXXXXXXXXXXXXXXXX
A. U User Defined      .      .GEN_CODE          .
- V View Defined       .      .HILFSDAT         .
S. 1 Parm Defined     .      .MAP              .
Z. 2 Local Defined     .      .OBJ              .
C.....XXXXXXXXXXXXX .XXXX. OBJECTTYPE      .
Street/Number: XXXXXXXXXXXXXXXX .PERSONNEL    .
                      .PERSONNEL-FILE      .
                      .....LIB= SYSTEM..

Take variable definition from view
    
```

Select the desired view (for example, PERSONNEL) and press ENTER. The selected view is displayed in a window:

```

*** Personnel Data Detail Display Function ***

Person Data                               Employment Data
-
Name      : XXXXXXXXXXXXXXXXXXXXXXXX      Job       : XXXXXXXXXXXXXXXXXXXXXXXX
Initial   : X                               Salary    : 999999
First-Name : XXXXXXXXXXXXXXXXXXXXXXXX      Commission: 999999
Children  : 99                               Years-Educ: 99
Family Status: XXXXXXXXXXXX                Years-Comp: 99
Sex       : X                               Vacation-D: 99
                                                Sick-Days : 99

                                                Hobby:    XXXXXXXXXXXXXXXXXXXXXXXX

Address
-
..PERSONNEL.....
. 1 AA PERSONNEL-NUMBER          N 8.0 D .
. 1 AA PERSONAL-NUMMER          N 8.0 D .
. 1 AA NUMERO-PERSONNEL         N 8.0 D .
.G 1 G1 PERSON                  .
. 2 BA NAME                      A 20 N D .
.....

HD=PERSONNEL/NUMBER
    
```

This window displays, for the highlighted field, three additional information lines at the bottom of the screen.

These lines display the following information for the current field:

- Edit mask
- Header
- Comments

The field name consists of the view name concatenated with the field name by a period, for example:

**personal.personnel-number**

Select the fields which are to be included in your map and press ENTER.

The library window, listing the selected field in the "Field=" line appears and you are prompted to "Position cursor and press Enter".

Position the cursor to the start of a field position and press ENTER.

Continue modifying fields as described under Creating a User-Defined Variable.

Repeat this procedure until all views required for your map have been completed.

**Note:**

The user view field name is used as the map field name for fields selected from a user view, preceded by the name of the view.

## Defining Fields for a Parameter or Local Data Definition

New parameters or local variables can be created or existing ones can be modified by selecting the item "Parm Defined" or "Local Defined" from the field editing list and pressing ENTER.

### Parameter Definitions

If you select the "Parm Defined" item, the following window appears:

```

..PARAMETER.....
.<CREATE>.....
. TEST1          L      001:003      .
. TEST2          L      001:003,001:004 .
. TEST3          L      001:003,001:004,001:005 .
.....
    
```

With this function, new parameters can be added and existing parameters can be modified.

If you want to modify an existing parameter (for example, TEST3), select it with your cursor and press ENTER. A window pops up prompting you to define the desired action: you can Edit the selected parameter, Delete it, or Cancel the function.

If you define the action Edit, the following window appears, in which you can edit the selected parameter:

```

..PARAMETER.....
.Name.....: TEST3
.Format...: L
.
.Dimension: 3
.
. Lower Bnds : Upper Bnds
.1. Index.: 1      3
.2. Index.: 1      4
.3. Index.: 1      5
.....
    
```

If you want to create a new parameter, select the <CREATE> option. The same window appears, but this time it is empty so that you can make the specifications for the parameter to be created.

When you leave the above window, a further window pops up prompting you whether you want to either save your modifications/specifications or cancel the function.

### Local Data Definitions

If you select the "Local Defined" item, the following window appears:

```

..LOCAL.....
.<CREATE>.....
. TESTA          L      001:003      .
. TESTB          L      001:003,001:004 .
. TESTC          L      001:003,001:004,001:005 .
.....
    
```

With this function, new local variables can be added and existing variables can be modified. Local variables can be used to pass values from one processing rule to another.

If you want to modify an existing local variable (for example, TESTC), select it with your cursor and press ENTER. A window pops up prompting you to define the desired action: you can Edit the selected variable, Delete it, or Cancel the function.

If you define the action Edit, the following window appears, in which you can edit the selected local variable:

```

..LOCAL.....
•Name.....: TESTC      .
•Format...: L          .
.                      .
•Dimension: 3          .
.      Lower Bnds   :   Upper Bnds   .
•1. Index.:          1             3   .
•2. Index.:          1             4   .
•3. Index.:          1             5   .
.....
    
```

If you want to create a new local variable, select the <CREATE> option. The same window appears, but this time it is empty so that you can make the specifications for the variable to be created.

When you leave the above window, a further window pops up prompting you whether you want to either save your modifications/specifications or cancel the function.

# Map Profile

This section describes the process of defining the map settings (profile) for a map.

The Map Profile is invoked by selecting "Prof. Map" from the "Ops. Map" selection list. The following screen is displayed:

```

.....
.
.              NATURAL MAP EDITOR (Esc to select field)
.
.Cr..Map Settings.....
.  .Format              Context
.  -----
Add..Page Size .....:  24
-  .Line Size .....:  79      WRITE Statement ....: N
Sta..Layout .....:
Zip..dynamic .....: N      Help Routine .....:
Cit..Zero Print .....: N    Help Parameter .....:
Str..Upper Case .....: Y    as field default ..: N
.  .Documentation Skip ....: N      Help Text .....: N
.  .Decimal Char ...: .      Position Line .....: 0      Column :   0
.  .Standard Keys ..: N      AutoRuleRank .....: 1
.  .Right Justify ..: Y
.  .Print Mode .....:      Filler Characters
.  -----
.  .Control Var ....:      Optional, Partial ..: _
.  .                  ..:      Required, Partial ..: _
.  .Field Sensitive: N      Optional, Complete : _
.  .                  ..:      Required, Complete : _
.....
Modify map profile
    
```

This map profile screen comprises Map Profile Settings and Filler Characters.

## Map Profile Settings

The following map profile settings can be used:

Entry	Explanation
<b>Page Size</b>	The number of map lines to be edited (1 - 250); if "Std Keys" (see below) is set to "Y", the number of lines is restricted to 3 - 250. For a map which is output with a WRITE statement (see the entry "WRITE Statement" in the Context column of the Map Settings), you specify the number of lines of the logical page output with the WRITE statement, not the map size. Thus, the map may be output several times on one page.
<b>Line Size</b>	The number of map columns to be edited (5 - 249).
<b>Layout dynamic</b>	The name of a map source definition which contains a predefined layout. <b>Y</b> - Specifies the layout to be dynamic. The dynamically used layout does not become a fixed part of the map at compilation time, but is executed at runtime. Thus, subsequent modifications of a layout map become effective for all maps using that layout map. If the layout map includes user-defined variables, you have to define these parameters in the map using the layout map. Input fields and modifiable fields in the layout map are not open at runtime. Parameters can be added by pressing F9 within the Field and Variable Definitions function. <b>N</b> - Specifies the layout to be static. The static layout is copied into the source area when a map is initialized. Filler characters are not transferred; "N" is the default setting.

Entry	Explanation
<b>Zero Print</b>	<p><b>Y</b> - displays a field value of all zeros as one zero only.</p> <p><b>N</b> - displays a zero value as blanks.</p> <p>This value is copied into the field definition when a new field is created and can be modified for individual fields using the extended field editing function.</p>
<b>Upper Case</b>	<p><b>Y</b> - indicates that all input entered for fields at map execution time is to be converted to upper case.</p> <p><b>N</b> - indicates that no lower to upper case conversion is to be performed.</p> <p>This value is copied into the field definition when a new field is created, and can be modified for individual fields using the extended field editing function.</p>
<b>Documentation Skip</b>	<p><b>Y</b> - Does <i>not</i> automatically move the cursor to the next field in the map at execution time even if the current field is completely filled.</p> <p><b>N</b> - Moves the cursor automatically to the next field in the map at execution time when the current field is completely filled; "N" is the default setting.</p>
<b>Decimal Char</b>	The character to be used as the decimal notation character. This character can only be changed with the GLOBALS command.
<b>Standard Keys</b>	<p><b>Y</b> - leaves the last two lines of the map empty so that function-key specifications can be entered at execution time.</p> <p><b>N</b> - causes all lines to be used for the map.</p>
<b>Right Justify</b>	<p>The type of field justification to be used for numeric and alphanumeric fields taken from a user view or the data definition:</p> <p><b>Y</b> - right justified</p> <p><b>N</b> - left justified</p>
<b>Print Mode</b>	<p>The default print mode for variables:</p> <p><b>C</b> - indicates that an alternative character set is to be used (special character table).</p> <p><b>I</b> - indicates inverse print direction.</p> <p><b>IC</b> - indicates standard print direction. This value is copied into the field definition when a new field is created.</p>
<b>Control Var</b>	The name of a control variable, the content of which determines the attribute characteristics of fields and texts that have the attribute definition AD=Y or (Y). The maximum length of a control variable is limited to 8 characters. The control variable referenced in the map must be defined in the program using that map.
<b>Field Sensitive</b>	<p><b>Y</b> - specifies that processing rules attached to map fields are executed on a field-by-field basis; that is, immediately after you have left a given field.</p> <p><b>N</b> - specifies that no field sensitivity is to be defined for the map, which means that the map is not to be processed until you have entered all necessary values in the map fields and pressed ENTER (or any F key).</p>
<b>WRITE Statement</b>	<p><b>Y</b> - Marking this field with <b>Y</b> produces a WRITE statement at the end of the map definition process. The resulting map may then be invoked from a Natural program using a WRITE USING FORM statement. Empty lines at the end of the map are automatically deleted so that the map can be output several times on one page.</p> <p><b>N</b> - Marking this field with <b>N</b> will cause the result of the map definition process to be an INPUT statement. The resulting map may then be invoked from a Natural program using an INPUT statement.</p>
<b>Help Routine</b>	The name of a help routine which is invoked at runtime when the help function is invoked for this map (global help for map). For detailed explanation of the syntax, see the parameter "Help" in the section Modifying a User-Defined Variable - Field Editing.

Entry	Explanation
<b>Help Parameter</b>	The help parameter which is invoked at execution time when the help function is invoked. <b>Note:</b> A maximum of 20 help parameters are possible. If you enter more, they are simply ignored.
<b>as field default</b>	<b>Y</b> - specifies that the help routine for the map is to apply as default to each individual field on the map, which means that the name of each field is passed individually to the help routine. <b>N</b> - specifies that the name of the map is passed to the help routine.
<b>Help Text</b>	<b>Y</b> - specifies that this map is actually help text; default = "N".
<b>Position Line Column</b>	The position where the help map is to appear on the screen at execution time.
<b>AutoRuleRank</b>	The rank (priority) assigned to automatic Predict rules when they are linked to the map during field definition. Default is 1.

## Filler Characters

Filler characters can be assigned to indicate whether information for a field is mandatory and whether the field must be completely filled:

Field Type	Explanation
<b>Optional Partial</b>	Is not mandatory, need not be completely filled.
<b>Required Partial</b>	Mandatory, need not be completely filled (AD=E).
<b>Optional Complete</b>	Is not mandatory, must be completely filled (AD=G).
<b>Required Complete</b>	Mandatory, must be completely filled (AD=EG).

Filler characters may also be defined for individual fields using the extended field editing function. For definition of field types, see also the session parameter AD.

## Post Assignment

A field which has been previously defined (in layout) directly on the screen may be assigned the field name and field attributes of a user view field or a DEFINE DATA definition.

**Note:**

Duplicate field names are only allowed for fields defined as "output only fields".

A map field which has been created using a DDM field definition may be redefined using the field definition from a view defined in a data area. This, however, is only possible if it is the same database field.

Post assignment can be done by entering the user view field number (or letter) as shown in the view window.

This function may only be used if the formats of the layout agrees with the field definition. N and P are considered to be identical numeric.

This function may not be used for view arrays if one or more dimensions of that array are smaller than the dimensions of the array in the layout.

## Field-Sensitive Processing

In the map profile, you can specify whether a map is to be processed after you have entered all necessary values in the map fields and pressed ENTER, or whether the processing rules attached to the map fields are to be handled on a field-by-field basis.

When handled on a field-by-field basis, the processing rules attached to a field are executed immediately as soon as you have filled this field entirely or you move the cursor to another position; they are executed in the same order as without field sensitivity.

### Advantages of Field-Sensitive Processing

The advantages of field-sensitive processing are:

- dynamic filling of fields based on user input,
- improved user guidance,
- security based on user input,
- rapid data entry.

**Note:**

To exploit the advantages of field-sensitive processing, you need to adapt your existing processing rules accordingly. For more information on processing rules, see Rule Editing - Processing Rules.

### Dynamic Fillings of Fields

Without field-sensitive processing, the application would wait until all the fields have been filled in and you have pressed ENTER, before checking if the data entered are valid. In the worst case, you would have wasted time and effort filling in all the fields, before being informed of the error.

However, with field-sensitive processing, as soon as you have entirely filled a field or leave the field by either pressing END or SHIFT+END or moving the cursor with one of the cursor movement keys, the data input is checked immediately, and other fields can be pre-filled automatically (via REINPUT FULL) with data based on the value entered in the previous field. Depending on this value, these data may also be the result of a database query.

The information under which condition a field was left can be retrieved from the \*PF-KEY system variable, which contains either "FULL" (if the field was entirely filled) or the name of the key most recently pressed as shown in the following table.

<b>Pressed Key</b>	<b>*PF-KEY</b>
LEFT ARROW	LEFT
RIGHT ARROW	RIGT
UP ARROW	UP
DOWN ARROW	DOWN
PAGE UP	PGUP
PAGE DOWN	PGDN
TAB	TAB
HOME	HOME
END	END
BACKTAB	BTAB
ENTER	ENTR
Field is full	FULL

### **Improved User Guidance**

With field-sensitive processing, you can be guided from field to field, depending on the values entered. The cursor can skip fields that are pre-filled and take you to the next input field.

As messages are displayed immediately after a field has been checked, you are informed more promptly and more precisely as to what to do.

### **Security Based on User Input**

Depending on the value entered in a particular field, you can be prompted, for example, for a special password to access the requested information.

### **Rapid Data Entry**

All these mechanisms described above make interaction with the application much faster and more efficient.

## **Defining a Map as Field-Sensitive**

To be able to use field-sensitive processing, you first need to define a map as being field-sensitive by setting Field Sensitive to "Y" in the Map Profile. A window appears asking you whether field sensitivity should be with or without automatic field recognition:

```

.....
.
              NATURAL MAP EDITOR (Esc to select field)
.
·Cr·Map Settings.....
  ·Format                Context                .
  -----
Add·Page Size .....: 24
- ·Line Size .....: 79      WRITE.....
Sta·Layout .....:          · Automatic Field Recognition ? ·
Zip·dynamic .....: N       Help ·          YES   NO   .
Cit·Zero Print ....: N     Help .....
Str·Upper Case ....: Y     as field default .: N
  ·Documentation Skip ...: N   Help Text .....: N
  ·Decimal Char ..: .       Position Line .....: 0   Column: 0
  ·Standard Keys .: N       AutoRuleRank .....: 1
  ·Right Justify .: Y
  ·Print Mode ....:        Filler Characters
  .
  ·Control Var ...:        Optional, Partial .: _
  .
  ·Field Sensitive: Y      Required, Partial .: _
                          Optional, Complete : _
                          Required, Complete : _
.....
Modify map profile
    
```

Field sensitivity within a Natural map can be achieved in two different ways:

1. With automatic field recognition, which means that you need not code any conditions for activating the right processing rule.

**Note:**

If you upload such a map to the mainframe, the Natural mainframe map editor does not understand the syntax of the uploaded map, because of new syntax extensions.

2. Without automatic field recognition, which means that you are responsible for being informed about which field was left with the last keystroke. You can achieve this by using the system variables \*CURS-COL and \*CURS-LINE in your Natural program.

**Note:**

If you upload such a map to the mainframe, STOW the map on the mainframe, and the same source code will be generated.

# SYSMAIN Utility

The SYSMAIN utility, which is available online and in batch mode, is used to transfer objects within the Natural system from one environment to another.

- Libraries
- Maintaining Libraries
  - List Function
  - Find Function
  - Copy, Move and Rename Functions
  - Delete Function
  - Import Function
- Invoking SYSMAIN by a Subprogram
  - Direct Commands
  - Additional Keywords for Direct Commands

# Libraries

Libraries are used to store Natural objects such as programs, subprograms, subroutines, help routines, and text.

Some libraries are delivered with Natural, such as libraries beginning with the prefix "SYS". The library SYSTEM, for example, contains system-related programs, error messages, and DDMs (data definition modules).

When you first logon to Natural, your default library will in most cases be "SYSTEM". Because this library is generally full of system-related objects, you will probably want to use a different library to store your programs. You can add your own libraries to the Natural environment or use libraries set up for this purpose.

**Note:**

You can alter your default library from "SYSTEM" to any other library by using the Natural profile parameter INIT-LIB (see Installing and Setting Up Natural on UNIX or Installing and Setting Up Natural on OpenVMS ).

To log on to a library, select "Library" on the Main Menu; a list appears showing the function <LOGON> followed by a list of all existing libraries:

```

2002-02-15                NATURAL                Library: SAG
14:34:19                  V 5.1.1 Software AG 2002      Mode   : STRUCTURED
User: SAG                  Work Area  : empty
.....
·Library      Direct      Services      OS      Fin      ·
.....
· <LOGON>    ·
· DEMO1     ·
· DEMO2     ·
· DEMO3     ·
· SAMPLES   ·
· SYSEXP   ·
· SYSEXRM   ·
· SYSTEM    ·
.....

Select Library
    
```

To create a new library, you select <LOGON>. A window is then displayed in which you enter the name of the library to be created. For information on naming conventions for libraries, see the system command LOGON.

To use an existing library, you either select the library from the list, or select <LOGON> and then enter the name of the desired library. The contents of the library are then displayed:

```

..... List * * .....
· Cmd Name      Type      SM S/C User ID  SRC Date      GP Date      ·
· -----
· <DIRECT COMMAND>
·   ARRAYD     Program    S  S   SAGPC    14:35 02-02-15
·   ARRAYE     Program    S  S/C SAGPC    14:35 02-02-15 11:25 02-02-18
·   BREAK1     Program    S  S/C SAGPC    14:35 02-02-15 11:25 02-02-18
    
```

Field	Explanation
<b>Cmd</b>	<p>In this field, you can enter a command code. To get a list of the possible commands, you press F2. Only the commands valid for the object selected are shown.</p> <p><b>C - Check:</b> Checks source program for syntax errors.</p> <p><b>D - Read:</b> Reads object's source program into work area.</p> <p><b>E - Edit:</b> Reads object's source program into work area.</p> <p><b>L - List:</b> Lists object's source program.</p> <p><b>H - Hardcopy:</b> Sends copy of object to printer.</p> <p><b>R - Run:</b> Compiles and executes source.</p> <p><b>S - Stow:</b> Stores object in source and object form.</p> <p><b>U - Scratch:</b> Deletes object in source and object form.</p>
<b>Name</b>	Object name.
<b>Type</b>	Object type.
<b>SM</b>	Programming mode (R = reporting mode, S = structured mode).
<b>S/C</b>	<p>S - The object exists in source form.</p> <p>C - The object exists in cataloged form.</p>
<b>User ID</b>	The ID of the user who saved/cataloged the object.
<b>SRC Date</b>	Source Program Date: The time and date when the object was last saved.
<b>GP Date</b>	Generated Program Date: The time and date when the object was last cataloged.

## Maintaining Libraries

The SYSMAIN utility is used to perform various maintenance functions on Natural objects within a library and across different libraries. The following objects can be maintained: Program, Subprogram, Subroutine, Map, Data Area (local, parameter, and global), Copycode, Helprountine, Recording, Dialog, ExpertModel, Text.

To invoke SYSMAIN, you select "Services" on the Natural Main Menu. A selection window will be displayed:

```

2002-02-15                NATURAL                Library: SAG
14:38:45                   V 5.1.1 Software AG 2002        Mode   : STRUCTURED
User: SAG                               Work Area : empty
.....
·Library      Direct      Services      OS      Fin      ·
.....
                        .....
                        ·D DDM Services      ·
                        ·S SYSMAIN          ·
                        .....
    
```

From this window, you select "SYSMAIN". A selection window with the following functions will be displayed:

Function	Description
<b>List</b>	Display a range of objects within a specific library.
<b>Find</b>	Locate a single object within a specific library.
<b>Copy</b>	Copy object from one library to another library.
<b>Move</b>	Transfer object from one library to another.
<b>Delete</b>	Delete object from a specific library.
<b>Rename</b>	Give an object a new name and optionally transfer it to a new library.
<b>Import</b>	Copy object from an external source into a Natural library.
<b>Terminate</b>	Leave SYSMAIN.

On the first screen displayed with each of the above functions (except Import), you can select either Programming Objects or DDMs for processing:

- To process DDMs, you mark the Views option with an "X" and press ENTER.
- To process another type of object, you mark the Programming Objects option with an "X" and press ENTER.

All functions are explained in the following sections and assume the selection of Programming Objects.

The view or programming object environment specification must always correspond to the relevant database ID (DBID) and file number (FNR) of an FNAT or FUSER system file.

### List Function

The List function is used to display the source code of a specific object. This object can be selected from a range of objects contained within a specific library in a specific environment.

If you select first the List function and then the Programming Objects option, the following window is displayed:

```

..... LIST .....
.
. OBJECT: *
. LIBRARY: SYSTEM          CODE: X (S)ource
. DBID: 99   FNR: 42       X (C)ataloged
. USER ID: SAGEX
. DATE:    - -           :
.....
    
```

In this window, you can specify the following items:

<b>Object</b>	The name of the object to be listed. You can use asterisk notation (*) to find all objects or those objects whose names begin with a specific character string.
<b>Library</b>	The name of the library containing the object to be listed. You can use asterisk notation (*) to list for selection all libraries or those libraries whose names begin with a specific character string. For example, for a list of all libraries whose names begin with "SY", you enter "SY*".
<b>DBID</b>	The number (ID) of the database which contains the object(s) to be listed.
<b>FNR</b>	The number of the Natural system file which contains the object(s) to be listed.
<b>User ID</b>	The user ID under which the object was most recently cataloged and/or saved. All objects cataloged and/or saved by the specified user are selected.
<b>Date</b>	The search begin date and time for cataloged/saved objects. All objects cataloged/saved after this date and time are selected.
<b>Code</b>	The object form (S = source, C = cataloged). Because it is possible to list the source code of an object, but not its cataloged code, the field "Source" must be selected. If only "Cataloged" is selected, all cataloged objects are listed, but none can be displayed.

If asterisk notation is used to specify the library and more than one library is found, then all libraries found are displayed for selection:

```

..... LIST .....
.  Library   DBID/FNR  .
. -----  .
. SYSEXP   99/42     .
. SYSEXR   99/42     .
. SYSTEM   99/42     .
. -----  .
. *** ENTER==>list *** .
. ***   ESC==>exit  *** .
.....
    
```

Select one library from the list and press ENTER. If asterisk notation is used to specify the object and more than one object is found, then object types are displayed for selection:

```

.....OBJECT TYPE.....
.X ==> select ALL .
. Program .
. Subroutine .
. Copycode .
. Map .
. Text .
. Helproutine .
. Subprogram .
. Global Data .
. Local Data .
. Parameter Data .
. Recording .
. Dialog .
. Class .
. ExpertModel .
.....
    
```

If you mark "ALL" or select the desired object type from the list, a list of either all objects or of all objects of the selected type contained in the previously selected library is displayed:

```

..... 10 Object(s) in Lib: SYSEXP.....
. Object   Type       S/C   User ID .
. -----  -
. BREAK1  Program      S/C   SAGEX .
. EMPL-LDA Local       S     SAGEX .
. IMP1    Program      S     SAGEX .
. IMP2    Program      S/C   SAGEX .
. IMP3    Program      S/C   SAGEX .
. LINKDEMO Program      S     SAGEX .
. LOGONQ-G Subprogram  S     SAGEX .
. NL-GDA  Global        S     SAGEX .
. -----  -
. *** ENTER==>list *** ESC==>exit *** .
.....
    
```

For each object listed, the object name, object type, object form (S = source, C = cataloged), and user ID under which the object was saved/cataloged are displayed.

To display the source code of an object, select the desired object from the list.

```

..... List: BREAK1 .....
. 0010 ***** .
. 0020 * EXAMPLE: 'BREAK1': AT BREAK STATEMENT .
. 0030 * .
. 0040 * PURPOSE: DEMONSTRATE NATURAL SYSTEM FUNCTIONS WITH AT BREAK .
. 0050 * CONDITION. INCLUDE USER-SUPPLIED TEXT. .
. 0060 * .
. 0070 * HIGHLIGHTS: AT BREAK STATEMENT, NATURAL SYSTEM FUNCTIONS OLD, MIN, .
. 0080 * AVER, MAX, SUM, TOTAL, COUNT .
. 0090 ***** .
. 0100 DEFINE DATA .
. 0110 LOCAL .
. 0120 1 EMPLOY-VIEW VIEW OF EMPLOYEES .
. 0130 2 NAME .
. 0140 2 CITY .
. 0150 2 SALARY (1) .
.....
    
```

To leave this function, you press ESC.

## Find Function

The Find function is used to determine in which library of which environment a specific object is contained. Once found, the object's source code can be displayed.

If you select the Find function first and then the Programming Objects option, the following window is displayed:

```

..... FIND .....
.
. OBJECT:  B*
. LIBRARY: SYSEXP          CODE: X (S)ource
. DBID:    99   FNR:    42      X (C)ataloged
. USER ID: SAGEX
. DATE:    -   -   :
.....
    
```

In this window, you can specify the following items:

<b>Object</b>	The name of the object you are looking for. You can use asterisk notation (*) to find all objects or those objects whose names begin with a specific character string. For example, for a list of all objects whose names begin with "B", you enter "B*".
<b>Library</b>	The name of the library in which you want to look for the object to be found. You can use asterisk notation (*) to search for all libraries or only those libraries whose names begin with a specific character string.
<b>DBID</b>	The number (ID) of the database which contains the object(s) you are looking for.
<b>FNR</b>	The number of the Natural system file which contains the object(s) you are looking for.
<b>User ID</b>	The user ID under which the object to be found was most recently cataloged and/or saved. All objects cataloged and/or saved by the specified user are selected.
<b>Date</b>	The search begin date and time for cataloged/saved objects. All objects cataloged/saved after this date and time are selected.
<b>Code</b>	The object form (S = source, C = cataloged). You can find objects which are available in either source or cataloged form only, or objects which are available in both source and cataloged form (default). <b>Note:</b> Only source objects can be displayed.

A list of objects is then displayed:

```

..... 4 Object(s) in Lib: SYSEXP.....
. Object   Type           S/C   User ID  .
. -----  - - - - - - - - - - - - - - - .
. BREAK1  Program          S/C   SAGEX   .
. BREAK2  Program           S     SAGEX   .
. BREAK3  Program           S     SAGEX   .
. BREAK4  Program          S/C   SAGEX   .
. -----  - - - - - - - - - - - - - - - .
. *** ENTER==>list *** ESC==>exit *** .
.....
    
```

For each object found, the object name, object type, object form (S = source, C = cataloged), and user ID under which the object was saved/cataloged are displayed.

To display the source code of an object, select the desired object from the list.

```

..... List: BREAK1 .....
. 0010 ***** .
. 0020 * EXAMPLE:      'BREAK1': AT BREAK STATEMENT .
. 0030 * .
. 0040 * PURPOSE:      DEMONSTRATE NATURAL SYSTEM FUNCTIONS WITH AT BREAK .
. 0050 *                CONDITION. INCLUDE USER-SUPPLIED TEXT. .
. 0060 * .
. 0070 * HIGHLIGHTS:  AT BREAK STATEMENT, NATURAL SYSTEM FUNCTIONS OLD, MIN, .
. 0080 *                AVER, MAX, SUM, TOTAL, COUNT .
. 0090 ***** .
. 0100 DEFINE DATA .
. 0110     LOCAL .
. 0120 1 EMPLOY-VIEW VIEW OF EMPLOYEES .
. 0130     2 NAME .
. 0140     2 CITY .
. 0150     2 SALARY (1) .
.....

```

To leave this function, you press ESC.

## Copy, Move and Rename Functions

The **Copy** function is used to copy objects from a library in a source environment to a library in a target environment. The objects in the original library are not deleted.

The **Move** function is used to move objects from a library in a source environment to a library in a target environment. A move is basically a copy operation in which the objects are deleted from the source library.

The **Rename** function is used to rename objects. You can either rename the object in the source environment or rename the object and transfer it to another (that is, target) environment. Since renaming implies deletion of the original object in the source environment, you are prompted with an option to retain it. If you decide to retain it, the original object is not modified.

For the copy, move, and rename functions, if the target environment already contains an object with the same name as the object to be copied, moved or renamed, the specified object is not copied, moved or renamed.

### Note:

Because of the similarities among the copy, move and rename functions, this section describes only the copy function.

Select the Copy function and then the Programming Objects option. The following window is displayed:

```

..... COPY .....
.
.           - Source -
. OBJECT:  *
. LIBRARY: SYSEXP          CODE: X (S)ource
. DBID:    99   FNR:    42      X (C)ataloged
. USER ID: SAGEX          XREF: n
. DATE:    - -      :
.
.           - Target -
. OBJECT:
. LIBRARY:                REPLACE:
. DBID:    FNR:
.....
    
```

In this window, you can enter the following source specifications:

<b>Object</b>	The name of the object to be copied. You can use asterisk notation (*) to list all objects or those objects whose names begin with a specific character string. You can then select specific objects from the resulting list for copying.
<b>Library</b>	The name of the library which contains the object(s) to be copied. You can use asterisk notation (*) to list all valid libraries or those libraries whose names begin with a specific character string. You can then select a specific library from the resulting list.
<b>DBID</b>	The number (ID) of the database which contains the object(s) to be copied.
<b>FNR</b>	The number of the Natural system file which contains the object(s) to be copied.
<b>User ID</b>	The user ID under which the object to be copied was most recently cataloged and/or saved. All objects cataloged and/or saved by the specified user are selected.
<b>Date</b>	The search begin date and time for cataloged/saved objects. All objects cataloged/saved after this date and time are copied.
<b>Code</b>	The object form (S = source, C = cataloged). You can copy objects which are available in either source or cataloged form only, or objects which are available in both source and cataloged form (default).
<b>XREF</b>	Indicates whether SYSMAIN is to support cross-reference data stored on Predict system files.  <b>NO</b> - Cross-reference data are not processed, except when using the DELETE function. If a cataloged object is deleted, SYSMAIN always deletes any existing XREF data for this object.  <b>YES</b> - All cross-reference data are processed.  <b>FORCE</b> - All cross-reference data are processed and the object must be documented in Predict. (Not yet implemented)  <b>DOC</b> - All cross-reference data are processed and the object must be documented in Predict but XREF data are not copied. (Not yet implemented)

If you either specify the desired library or select it from the list and use asterisk notation (\*) in the Object field, a window is displayed, which lists all available object types:

```

.....OBJECT TYPE.....
.X ==> select ALL .
. Program .
. Subroutine .
. Copycode .
. Map .
. Text .
. Helproutine .
. Subprogram .
. Global Data .
. Local Data .
. Parameter Data .
. Recording .
. Dialog .
. Class .
. ExpertModel .
.....
    
```

Mark "ALL" to process all object types or mark a specific object name to process only objects of this type. In the window below, all objects have been listed:

```

..... Select for COPY in Lib: SYSEXPG.....
. X Object      Type          S/C  User ID .
. - - - - - - - - - - - - - - - - - - - - .
.   ARRAYD     Program        S/C  SAGEX   .
.   ARRAYE     Program        S    SAGEX   .
. X BREAK1     Program        S/C  SAGEX   .
. X BREAK2     Program        S    SAGEX   .
. X BREAK3     Program        S    SAGEX   .
.   BREAK4     Program        S/C  SAGEX   .
.   EDITWORK   Subprogram    S    SAGEX   .
.   GDA01      Global          S    SAGEX   .
.   HELP01     Map            S    SAGEX   .
.   HOLDLOG    Program        S    SAGEX   .
.   LDA01      Local          S    SAGEX   .
.   MAP001X    Map            S    SAGEX   .
.   MAP002X    Map            S    SAGEX   .
.....
    
```

For each object listed, the object name, object type, object form (S = source, C = cataloged), and user ID under which the object was saved/cataloged are displayed.

You can select objects by marking them with an "X".

Once you have selected the objects, the following screen is displayed (for Copy and Move functions only; not applicable for Rename function):

```

.....
.X Select the specified Object(s) for copy
. Copy ALL specified Object(s)
.....
    
```

If you select the first function, a window is displayed for each selected object in which you can specify the new name to be used for the target object.

If you select the second function, all selected objects are copied/moved using the same object name for both the source and target object.

### Specifying the Target Environment for Copy/Move

```

..... COPY .....
.
.   3 Object(s) selected
.
.           - Source -
. OBJECT:  *
. LIBRARY: SYSEXP          CODE: X (S)ource
. DBID:   99   FNR:   42       X (C)ataloged
. USER ID: SAGEX           XREF: n
. DATE:   - -           :   TYPE: all
.
.           - Target -
. OBJECT:  *
. LIBRARY: SYSTEST         REPLACE: n
. DBID:   30   FNR:   20
.....
    
```

Enter the following target specifications; the objects are then copied/moved accordingly:

<b>Object</b>	Name(s) for the target object(s).
<b>Library</b>	The name of the library which is to contain the copied or moved object(s).
<b>Replace</b>	<p>Replace option for objects moved or copied. It can be used to overwrite the object in the target environment; the following values can be specified:</p> <p><b>Y</b> - An object with the same name which is already present in the target library is to be replaced.</p> <p><b>N</b> - An object with the same name which is already present in the target library is not to be replaced; "N" is the default.</p> <p><b>Note:</b> If a programming object is replaced, it is also deleted from the Natural buffer pool.</p>
<b>DBID / FNR</b>	The number (ID) of the database and the file number of the Natural system file into which the copied or moved object is to be placed.

### Specifying the Name and Target Environment for Rename

```

..... RENAME .....
.
.           - Old object name -
. OBJECT:  BREAK1
. LIBRARY: SYSEXP          CODE: X (S)ource
. DBID:   99   FNR:   42       X (C)ataloged
. USER ID: SAGEX           XREF: n
. DATE:   - -           :
.
.           - New object name -
. OBJECT:  PROG02
. LIBRARY: SYSEXP          REPLACE: n
. DBID:   99   FNR:   42
.....
    
```

To rename one or more objects, enter the following specifications; the object is then renamed:

<b>Object</b>	New object name. By using asterisk notation (*), you can rename multiple objects whose names begin with the same character(s). For example, if you have selected objects whose names begin with "ABC" and want them all to begin with "XYZ", you enter "XYZ*" as new name. The remainder of each name (after the first three characters) is retained.
<b>Library</b>	The name of the library to contain the renamed object(s).
<b>Replace</b>	Replace option for renamed object which is being renamed. It can be used to overwrite the object in the target environment. Values: <b>Y</b> - An object with the same name which already exists in the target library is to be replaced. <b>N</b> - An object with the same name which already exists in the target library is not to be replaced; "N" is the default. <b>Note:</b> If a programming object is replaced, it is also deleted from the Natural buffer pool.
<b>DBID / FNR</b>	The number (ID) of the database and the file number of the Natural system file into which the renamed object is to be placed.

In the bottom half of the window, specify the new object name(s) and target environment (if applicable).

### Delete Function

The Delete function is used to delete one or more objects from a library.

If you select the Delete function first and then the Programming Objects option, the following window is displayed:

```

..... DELETE .....
.
. OBJECT:  BR*
. LIBRARY: SYSEXPG          CODE: X (S)ource
. DBID:    99  FNR:  42      X (C)ataloged
. USER ID: SAGEX           XREF: n
. DATE:    - -             : CONFIRM: n
.....
    
```

Enter the following specifications for the object(s) to be deleted:

<b>Object</b>	The name of the object to be deleted. You can use asterisk notation (*) to list all objects or those objects whose names begin with a specific character string. You can then select specific objects from the resulting list for deleting.
<b>Library</b>	The name of the library which contains the object(s) to be deleted. You can use asterisk notation (*) to list all valid libraries or those libraries whose names begin with a specific character string. You can then select a specific library from the resulting list.
<b>DBID</b>	The number (ID) of the database which contains the object(s) to be deleted.
<b>FNR</b>	The number of the Natural system file which contains the object(s) to be deleted.
<b>User ID</b>	The user ID under which the object to be deleted was most recently cataloged and/or saved. All objects cataloged and/or saved by the specified user are selected.
<b>Date</b>	The search begin date and time for cataloged/saved objects. All objects cataloged/saved after this date and time are deleted.
<b>Code</b>	The object form (S = source, C = cataloged). You can delete objects which are available in either source or cataloged form only, or objects which are available in both source and cataloged form (default).
<b>XREF</b>	Indicates whether SYSMAIN is to support cross-reference data stored on Predict system files.  <b>NO</b> - Cross-reference data are not processed, except when using the DELETE function. If a cataloged object is deleted, SYSMAIN always deletes any existing XREF data for this object.  <b>YES</b> - All cross-reference data are processed.  <b>FORCE</b> - All cross-reference data are processed and the object must be documented in Predict. (Not yet implemented)  <b>DOC</b> - All cross-reference data are processed and the object must be documented in Predict but XREF data are not deleted. (Not yet implemented)

A window appears and you can then delete all objects or invoke a list displaying the objects to be deleted. You can then select specific objects from the list for deletion.

You select objects by marking them with an "X".

Once you have selected the objects, you are prompted to confirm the deletion.

## Import Function

The Import function is used to import objects which are currently located in another OpenVMS or UNIX directory to a Natural library.

Also, an object may be located in the correct directory, however is not recognized by Natural because it is not included in the library information file "FILEDIR.SAG". In this case, the object has to be imported.

When you invoke the Import function, the following window is displayed:

```

..... IMPORT .....
.
.           - Source -
.
. PATH:
. OBJECT:*          CODE: X (S)ource
.                   X (C)ataloged
.
.           - Target -
.
. LIBRARY:          USER ID:          REPLACE:
. DBID:             MODE:
. FNR:
.....
    
```

Enter the following Source specifications for the object(s) to be imported:

<b>Path</b>	The complete RMS or UNIX path name of the directory from which the Import function is to be executed. <b>Note for OpenVMS:</b> The path name may contain logicals. <b>Note for UNIX:</b> The path name may start with a UNIX environment variable; press ENTER and the environment variable is replaced by the full path name.
<b>Object</b>	The name(s) of the object(s) to be imported. You can use <b>asterisk notation (*)</b> to list all objects or those objects whose names begin with the same character(s). The objects to be imported can then be selected from the resulting list. <b>Note:</b> The object must have a valid extension to be recognized by SYSMAN.
<b>Code</b>	The object form (S = source, C = cataloged). You can import source code or cataloged code for an object or both source code and cataloged code (default).

A window is then displayed, from which you select the desired objects.

Once you have selected the objects to be imported, the following window is displayed, in which you specify the target information:

```

..... IMPORT .....
. 29 Object(s) selected
.
.           - Source -
.
. PATH:    $HOME
. OBJECT:  *          CODE:
. TYPE:
.
.           - Target -
.
. LIBRARY: SYSEXP    USER ID: SAGEX    REPLACE: n
. DBID:    99        MODE:    X Structured
. FNR:    42         Report
.....
    
```

Enter the following Target specifications:

<b>Library</b>	The Natural library into which the objects are to be imported. If the library does not already exist, a new library is created. You can use asterisk notation (*) to list all existing libraries.
<b>User ID</b>	The user ID under which the imported object is to be saved and/or cataloged.
<b>Replace</b>	<p>Replace option for an object which is being imported. It can be used to overwrite an object in the Target environment; the following values can be specified:</p> <p><b>Y</b> - An object with the same name which is already present in the target library is to be replaced.</p> <p><b>N</b> - An object with the same name which is already present in the target library is not to be replaced; "N" is the default.</p> <p><b>Note:</b> If a programming object is replaced, it is also deleted from the Natural buffer pool.</p>
<b>DBID / FNR</b>	The number (ID) of the database and the file number of the Natural system file into which the imported object is to be placed.
<b>Mode</b>	The programming mode. For Natural source objects, the options Structured (structured mode) and Report (reporting mode) are available. If you specify Structured, the programming mode of the objects being imported is set to structured mode.

Once you have entered the above information, the objects are imported into the specified library.

## Invoking SYSMAIN by a Subprogram

MAINUSER is a subprogram which allows you to perform the various SYSMAIN functions directly from any user-written object (subroutine, program or subprogram) without going through the normal steps of invoking SYSMAIN. Upon completion of processing of the SYSMAIN functions, the utility is terminated and control is returned to the object from which the request was issued. MAINUSER can be used in either online or batch mode.

MAINUSER is invoked as follows:

**CALLNAT 'MAINUSER' *command error message library***

The parameters are:

<i>command</i> (A250)	The direct command string to be executed by SYSMAIN.
<i>error</i> (N4)	The return code issued by SYSMAIN at the end of processing to indicate a normal end of processing or an error.
<i>message</i> (A72)	The message corresponding to the error given online.
<i>library</i> (A8)	The name of the library containing the utility SYSMAIN; by default, this is the library SYSMAIN. (Under UNIX, OpenVMS and Windows, this parameter does not apply and is provided for compatibility reasons only.)

An example of a callable routine is program MAINCALL in library SYSMAIN.

## Direct Commands

SYSMAIN functions can be executed using direct commands issued as a parameter of the MAINUSER subprogram.

Direct commands consist of keywords and parameters. The sequence of the direct command syntax is not completely fixed. The rules which apply are:

- Function, object type and object name must be the first three parameters of the command string.
- The library or path name must be specified immediately after the FROM, IN and TO keywords. (If the optional keyword LIBRARY or PATH is used, it must be entered between the FROM, IN or TO keyword and the library or path name).
- The WHERE clause must always follow the FROM, IN or TO keyword and the library name; the sequence of the keywords and values within the clause can be specified in any order.
- The keywords and values of the WITH clause can be specified in any order, but the WITH clause must always be placed at the end of the command string.

### Note:

In the syntax diagrams below, FM is shown instead of FROM to make the diagrams easier to read; however, FROM can always be used as a synonym for FM and vice versa.

## FIND and LIST Direct Command Syntax

The direct command syntax of the FIND and LIST functions is:

<b>{</b> <u>F</u> <b>I</b> <b>N</b> <b>D}</b> <b>{</b> <u>L</u> <b>I</b> <b>S</b> <b>T}</b>	<b>ALL</b>	<b>name</b> <b>IN</b> [ <b>LIBRARY</b> ] <i>lib-name</i> [ <i>where-clause</i> ] [ <i>with-clause</i> ]
	<u>C</u> <b>A</b> <b>T</b> <b>A</b> <b>L</b> <b>O</b> <b>G</b> <b>E</b> <b>D</b>	
	<u>S</u> <b>A</b> <b>V</b> <b>E</b> <b>D</b>	
	<u>S</u> <b>T</b> <b>O</b> <b>W</b> <b>E</b> <b>D</b>	
	<u>V</u> <b>I</b> <b>E</b> <b>W</b>	

The *where-clause* is optional. The syntax is:

<b>[</b> <b>WHERE</b> <b>]</b> [ <b>DBID</b> <i>dbid</i> ] [ <b>FNR</b> <i>file-nr</i> ]
--

The *with-clause* is optional. The syntax is:

<b>[</b> <b>WITH</b> <b>]</b> [ <b>TYPE</b> <i>type</i> ] [ <b>USER</b> <i>user-id</i> ] [ <b>XREF</b> <i>xref</i> ] [ <b>REPLACE</b> ] [ <b>RCOP</b> ] [ <b>NOPROMPT</b> ]
---

Since the WHERE clause and WITH clause syntax are identical for each function, they are only shown once with the FIND and LIST command syntax above.

#### Examples:

```
FIND PROG1 IN DBID 1 FNR 6
FIND STOWED MAINMENU IN SYS* WHERE DBID 1 FNR 5
FIND ALL PROG2 IN PROD* FNR 27 DBID 1
```

```
LIST VIEW * IN lib-name
L SAVED TEST* IN lib-name TYPE PNS FNR 6
L SA TEST* TYPE PM IN lib-name FNR 6 DBID 2
```

### COPY and MOVE Direct Command Syntax

The direct command syntax of the COPY and MOVE functions is:

<b>{</b> <u>C</u> <b>O</b> <b>P</b> <b>Y</b> <b>{</b> <u>M</u> <b>O</b> <b>V</b> <b>E</b> <b>}</b>	<b>ALL</b>	<b>name</b> <b>FM</b> [ <b>LIBRARY</b> ] <i>lib-name</i> [ <i>where-clause</i> ] <b>TO</b> [ <b>LIBRARY</b> ] <i>lib-name</i> [ <i>where-clause</i> ] [ <i>with-clause</i> ]
	<u>C</u> <b>A</b> <b>T</b> <b>A</b> <b>L</b> <b>O</b> <b>G</b> <b>E</b> <b>D</b>	
	<u>S</u> <b>A</b> <b>V</b> <b>E</b> <b>D</b>	
	<u>S</u> <b>T</b> <b>O</b> <b>W</b> <b>E</b> <b>D</b>	
	<u>V</u> <b>I</b> <b>E</b> <b>W</b>	

#### Examples:

```
COPY PROG1 FM TESTORD TO ORDERS DBID 1 FNR 6 REP
C PGM* WITH REP TYPE PNS FM TESTLIB TO PRODLIB
COPY VIEW PERS FM OLDLIB FNR 10 TO NEWLIB FNR 16 REPLACE
```

```
M VIEW PERSONNEL FM OLDLIB FNR 20 TO NEWLIB FNR 24
M PROG1 TO NEWLIB
MOVE STOWED * FM OLDLIB WITH XREF Y TO NEWLIB WHERE DBID 100 FNR 160
```

### DELETE Direct Command Syntax

The direct command syntax of the DELETE function is:

<b>DELETE</b>	$\left[ \begin{array}{c} \text{ALL} \\ \text{CATALOGED} \\ \text{SAVED} \\ \text{STOWED} \\ \text{VIEW} \end{array} \right]$	<i>name</i> [ IN [ <u>LIBRARY</u> ] <i>lib-name</i> [ <i>where-clause</i> ] ] [ <i>with-clause</i> ]
---------------	--	--

#### Examples:

```
D SA * IN LIBTEST TYPE GLA
DEL * TYPE PM IN TESTORD
DEL VIEW FINANCE IN TESTLIB DBID 12 FNR 27
```

### RENAME Direct Command Syntax

The direct command syntax of the RENAME function is:

<b>RENAME</b>	$\left[ \begin{array}{c} \text{ALL} \\ \text{CATALOGED} \\ \text{SAVED} \\ \text{STOWED} \\ \text{VIEW} \end{array} \right]$	<i>name</i> AS <i>new-name</i> FM [ <u>LIBRARY</u> ] <i>lib-name</i> [ <i>where-clause</i> ] TO [ <u>LIBRARY</u> ] <i>lib-name</i> [ <i>where-clause</i> ] [ <i>with-clause</i> ]
---------------	--	---

#### Examples:

```
RENAME PGM1 AS PROG1
REN PGM1 AS PROG1 FM TESTLIB DBID 1 FNR 5 TO PRODLIB DBID 2 FNR 6
```

### IMPORT Direct Command Syntax

The direct command syntax of the IMPORT function is:

<b>IMPORT</b>	$\left[ \begin{array}{c} \text{ALL} \\ \text{CATALOGED} \\ \text{SAVED} \\ \text{STOWED} \\ \text{VIEW} \end{array} \right]$	<i>name</i> { <u>FM</u> / <u>IN</u> } [ <u>PATH</u> ] <i>path-name</i> TO [ <u>LIBRARY</u> ] <i>lib-name</i> [ <i>where-clause</i> ] [ <i>with-clause</i> ]
---------------	--	--

**Examples for OpenVMS:**

```
IMPORT PGM1 FM HOME TO USERLIB
```

```
I PGM2 IN DISK:[USER.Natural.SRC.example.test] TO LIB TESTUSER
```

**Examples for UNIX:**

```
IMPORT PGM1 FM $HOME TO USERLIB
```

```
I PGM2 IN /usr/natural/src/example/test TO LIB TESTUSER
```

## Additional Keywords for Direct Commands

In addition to the keywords shown with the parameters above, the following keywords can also be used with direct commands to specify selection criteria:

Keywords	Explanation
ALL	All saved and/or cataloged programming objects are selected for processing.
CAT	All cataloged programming objects are selected for processing. (Any corresponding saved programming object is not processed.)
HELP	Activates online help.
IN/FM	Refers to a source environment.
<u>N</u> OPROMPT	Suppresses all prompts.
<u>R</u> COP	Used with direct commands to specify that a copy of the object being renamed is to be made.
<u>S</u> AVED	All saved programming objects are selected for processing. (Any corresponding cataloged object is not processed.)
STOWED	All programming objects which are both saved and cataloged are selected for processing.
TO	Refers to a target environment.
WITH	Optional keyword to indicate the start of a <i>with-clause</i> .
WHERE	Optional keyword to indicate the start of a <i>where-clause</i> .
.	End of command. If this character is detected anywhere within a command string, all subsequent data are ignored.